XVII Jornadas de Ingeniería del Software y Bases de Datos

Sistedes 2012

ACTAS

JISBD | PROLE | JCIS

Sistedes
Sociedad de Ingeniería de Software y Tecnologías de Desarrollo de Software

IN LUMINE SAPIENTIA · UNIVERSITAS ALMERIENSIS ·

Almería, 17 al 19 de Septiembre

Editores: Antonio Ruíz | Luis Iribarne

A. Ruíz, L. Iribarne (Eds.): Actas de las "XV*II Jornadas de Ingeniería del Software y Bases de Datos (JISBD'2012)*", Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

JISBD 2012

XVII Jornadas de Ingeniería del Software y Bases de Datos (JISBD)

Almería, 17 al 19 de Septiembre de 2012

Editores:
Antonio Ruíz
Luis Iribarne

Actas de las "*XVII Jornadas de Ingeniería del Software y Bases de Datos (JISBD)*"
Almería, 17 al 19 de Septiembre de 2012
Editores: Antonio Ruíz y Luis Iribarne
http://sistedes2012.ual.es
http://www.sistedes.es

ISBN: 978-84-15487-28-9
Depósito Legal: AL 674-2012

Prólogo

Las XVII Jornadas de Ingeniería del Software y Bases de Datos (JISBD) (JISBD 2012) se celebraron del 17 al 19 de Septiembre de 2012 en Almería y fueron organizadas por Grupo de Investigación de Informática Aplicada de la Universidad de Almería. Al igual que en anteriores ediciones, JISBD se celebró en paralelo y compartiendo algunos actos de las XII Jornadas de Programación y Lenguajes (PROLE) y de las VIII Jornadas de Ciencia e Ingeniería de Servicios (JCIS). Lo tres eventos son organizados bajo el auspicio de SISTEDES, la Sociedad de Ingeniería del Software y Tecnologías de Desarrollo de Software.

JISBD se ha consolidado como un foro de referencia donde investigadores y profesionales de España, Portugal e Iberoamérica, en los campos de la Ingeniería del Software y de las Bases de Datos, pueden debatir e intercambiar ideas, crear sinergias y, sobre todo, conocer la investigación que se está llevando a cabo en dicha comunidad. A fin de conseguir de manera efectiva este espacio de intercambio, las jornadas se organizaron por sesiones temáticas en las que han tenido cabida hasta cinco tipos de contribuciones: (1) trabajos regulares, que presentan algún resultado de investigación, (2) trabajos emergentes, que están comenzando su andadura, (3) demostraciones de herramientas, (4) trabajos relevantes ya publicados y (5) tutoriales. Para iniciar el debate indicando los aspectos más destacables y los más discutibles de cada contribución, los coordinadores de sesión delegaron parcialmente dicha responsabilidad en la figura del contraponente de cada contribución.

Las sesiones temáticas de esta edición han sido:

- *Sesión 1:* Bases de Datos, Almacenes de Datos, Minería de Datos, Recuperación de la información
- *Sesión 2:* Ingeniería Web, Interfaces de Usuario, Sistemas Colaborativos, Computación Ubicua
- Sesión 3: Apoyo a la decisión en Ingeniería del Software, Metodologías, Experimentación
- *Sesión 4:* Calidad, Pruebas y Requisitos
- *Sesión 5:* Desarrollo de Software Dirigido por Modelos
- *Sesión 6:* Líneas de Producto, Componentes y Arquitecturas Software
- *Sesión 7:* Otros aspectos de Ingeniería del Software y Bases de Datos.

Este volumen presenta las 86 contribuciones que han formado parte de esta edición: 35 trabajos regulares (con un 71% de ratio de aceptación), 19 trabajos emergentes (con un 89% de ratio de aceptación), 18 trabajos ya publicados, 14 herramientas y 2 tutoriales. También ofrece una breve reseña de la charla invitada impartida por el profesor Armando Fox de la Universidad de California, Berkeley titulada: “Cruzando el abismo educativo” de la ingeniería de software utilizando Software como Servicio y computación en nube. Agradezco que aceptara formar parte de estas Jornadas y su más que colaborativa disposición.

Un signo que acompaña la madurez de la comunidad es la existencia de un abanico de herramientas software cada vez más poblado y de mayor calidad. En esta edición se dispuso un comité de apoyo para su revisión y se organizó una breve sesión plenaria el último día donde dar a conocer y discutir sobre el “mapa de herramientas” de la comunidad JISBD. Estamos convencidos de que esta iniciativa aumentará las sinergias entre los grupos de investigación y por ende aumentará el valor del conocimiento científico y tecnológico que va atesorando nuestra comunidad.

Me gustaría expresar mi más sincero agradecimiento a los miembros del Comité de Programa por su tiempo y dedicación a la hora de revisar y seleccionar los artículos que fueron finalmente aceptados para su presentación, y que han permitido confeccionar un año más un programa de gran calidad y nivel. También a los distintos Coordinadores que se han ocupado de organizar aspectos esenciales como las demostraciones de herramientas (Cristina Vicente y Fernando Sánchez), trabajos relevantes (Amador Durán), tutoriales (Ángeles Saavedra) y coordinadores de las diferentes sesiones temáticas. Por supuesto, mi agradecimiento a los autores que enviaron artículos a las Jornadas, hayan sido aceptados o no, por su esfuerzo y contribución al evento.

También me gustaría agradecer al equipo del comité de organización liderado por Luis Iribarne su gran esfuerzo y excelente trabajo, que han permitido hacer realidad esta conferencia; al Comité Permanente de las JISBD por depositar su confianza a la hora de presidir el Comité de Programa, y por su constante apoyo y soporte. Mención especial merece Coral Calero, cuyos consejos y ayuda como presidente saliente han sido siempre inestimables. Un especial agradecimiento a la Universidad de Almería, que ha hecho posible que la conferencia fuera todo un éxito. Asimismo, este evento no hubiera sido posible sin el aval de la Sociedad de Ingeniería del Software y Tecnologías de Desarrollo de Software (SISTEDES) y sin la colaboración de la Asociación de Técnicos de Informática (ATI), y la oficina española del W3C.

Muchas gracias a todos los asistentes y participantes a las JISBD 2012, y esperamos verles de nuevo en las próximas JISBD.

Almería, Septiembre 2012

Antonio Ruiz-Cortés
Presidente del Comité de Programa de JISBD 2012

Prologo de la Organización

Las jornadas SISTEDES 2012 son un evento científico-técnico nacional de ingeniería y tecnologías del software que se celebra este año en la Universidad de Almería durante los días 17, 18 y 19 de Septiembre de 2012, organizado por el Grupo de Investigación de Informática Aplicada (TIC-211). Las Jornadas SISTEDES 2012 están compuestas por las XVII Jornadas de Ingeniería del Software y de Bases de Datos (JISBD'2012), las XII Jornadas sobre Programación y Lenguajes (PROLE'2012), y la VIII Jornadas de Ciencia e Ingeniería de Servicios (JCIS'2012). Durante tres días, la Universidad de Almería alberga una de las reuniones científico-técnicas de informática más importantes de España, donde se exponen los trabajos de investigación más relevantes del panorama nacional en ingeniería y tecnología del software. Estos trabajos están auspiciados por importantes proyectos de investigación de Ciencia y Tecnología financiados por el Gobierno de España y Gobiernos Regionales, y por proyectos internacionales y proyectos I+D+i privados. Estos encuentros propician el intercambio de ideas entre investigadores procedentes de la universidad y de la empresa, permitiendo la difusión de las investigaciones más recientes en ingeniería y tecnología del software. Como en ediciones anteriores, estas jornadas están auspiciadas por la Asociación de Ingeniería del Software y Tecnologías de Desarrollo de Software (SISTEDES).

Agradecemos a nuestras entidades colaboradoras, Ministerio de Economía y Competitividad (MINECO), Junta de Andalucía, Diputación Provincial de Almería, Ayuntamiento de Almería, Vicerrectorado de Investigación, Vicerrectorado de Tecnologías de la Información (VTIC), Enseñanza Virtual (EVA), Escuela Superior de Ingeniería (ESI/EPS), Almerimatik, ICESA, Parque Científico-Tecnológico de Almería (PITA), IEEE España, Colegio de Ingenieros Informática de Andalucía, Fundación Mediterránea, y a la Universidad de Almería por el soporte facilitado. Asimismo a D. Félix Faura, Director de la Agencia Nacional de Evaluación y Prospectiva (ANEP) de la Secretaría de Estado de I+D+i, Ministerio de Economía y Competitividad, a D. Juan José Moreno, Catedrático de la Universidad Politécnica de Madrid, presidente de la Sociedad de Ingeniería y Tecnologías del Software (SISTEDES), a D. Francisco Ruiz, Catedrático de la Universidad de Castilla-La Mancha, y a D. Miguel Toro, Catedrático de la Universidad de Sevilla, por su participación en la mesa redonda "*La investigación científica informática en España y el año Turing*"; a Armando Fox de la Universidad de Berkley (EEUU) y a Maribel Fernández del King's College London (Reino Unido), como conferenciantes principales de las jornadas, y a los presidentes de las tres jornadas por facilitar la confección de un programa de *Actividades Turing*. Especial agradecimiento a los voluntarios de las jornadas SISTEDES 2012, estudiantes del Grado de Ingeniería Informática y del Postgrado de Doctorado de Informática de la Universidad de Almería, y a todo el equipo del Comité de Organización que han hecho posible con su trabajo la celebración de una nueva edición de las jornadas JISBD'2012, PROLE'2012 y JCIS'2012 (jornadas SISTEDES 2012) en la Universidad de Almería.

Luis Iribarne
Presidente del Comité de Organización
@sistedes2012{JISBD;PROLE;JCIS}

Comité Científico

Presidente del Comité de Programa:

Antonio Ruiz Cortés (Universidad de Sevilla)

Coordinadores de Demostraciones:

Cristina Vicente-Chicote (Univ. Politécnica de Cartagena)
Fernando Sánchez (Univ. Extremadura)

Coordinadora de Tutoriales:

Ángeles Saavedra Places (Univ. A Coruña)

Coordinador de Divulgación de Trabajos Relevantes ya Publicados:

Amador Durán (Univ. de Sevilla)

Coordinadores de Sesiones Temáticas:

Coordinadores Sesión Temática 1:

Alfredo Goñi (Univ. País Vasco)
José Francisco Aldana (Univ. de Málaga).

Coordinadores Sesión Temática 2:

Pascual González (Univ. Castilla-La Mancha)
Juan Carlos Preciado (Univ. Extremadura)

Coordinadores Sesión Temática 3:

Mercedes Ruiz (Univ. Cádiz)
Agustín Yagüe (Univ. Politécnica de Madrid)

Coordinadores Sesión Temática 4:

Xavier Franch (Univ. Politécnica de Catalunya)
Claudio de la Riva (Univ. Oviedo)

Coordinadores Sesión Temática 5:

Antonio Vallecillo (Univ. Málaga)
José Raúl Romero (Univ. Córdoba)

Coordinadores Sesión Temática 6:

Carlos Canal (Univ. Málaga)
Silvia Abrahão (Univ. Politécnica Valencia)

Coordinadores Sesión Temática 7:

Coral Calero (Univ. Castilla-La Mancha)

Comité de Programa:

Ambrosio Toval (Univ. Murcia)
Ana María Moreno (Univ. Polit. Madrid)
Ana Moreira (Univ. Nova Lisboa)
Antonio Polo (Univ. Extremadura)
Antonio Rito (Univ. Tec. Lisboa)
Arantza Illarramendi (Univ. País Vasco)
Arantza Irastorza (Univ. País Vasco)
Artur Boronat (Univ. Leicester)
Carles Farré (Univ. Polit. Catalunya)
Carme Quer (Univ. Polit. Catalunya)
Cristina Cachero (Univ. Alicante)
Daniel Rodríguez (Univ. Alcalá)
David Benavides (Univ. Sevilla)
Dolors Costal (Univ. Polit. Catalunya)
Eduardo Fdez-Medina (Univ. Castilla-La Man)
Emilio Insfrán (Univ. Polit. Valencia)
Ernest Teniente (Univ. Polit. Catalunya)
Ernesto Pimentel (Univ. Málaga)
Esther Guerra (Univ. Autónoma de Madrid)
Félix García (Univ. Castilla-La Mancha)
Francisco Gutiérrez-Vela (Univ. Granada)
Francisco Ruiz (Univ. Castilla-La Mancha)
Goiuria Sagardui (Univ. Mondragón)
Ignacio Panach (Univ. Valencia)
Irene Garrigós (Univ. Alicante)
Isidro Ramos (Univ. Polit. Valencia)
Ismael Sanz (Univ. Jaume I)
Jaime Gómez (Univ. Alicante)
Javier Cámara (Univ. De Coimbra)
Javier Dolado (Univ. País Vasco)
Javier Jaén (Univ. Polit. Valencia)
Javier Tuya (Universidad de Oviedo)
Jenifer Pérez (Univ. Polit. Madrid)
Jesús García Molina (Univ. Murcia)
Jesús Torres (Univ. Sevilla)
Jesús Aguilar (Univ. Pablo Olavide)
Joan Fons (Univ. Polit. Valencia)
Joao Araujo (Univ. Nova Lisboa)
João Falcão e Cunha (Univ. Porto)
Jon Iturrioz (Univ. País Vasco)
Jordi Cabot (École des Mines de Nantes)
José Hilario Canós (Univ. Polit. Valencia)
José Luis Arjona (Univ. Huelva)
José Luis Fernández-Alemán (Univ. Murcia)
José Luis Roda (Univ. La Laguna)
José María Cavero (Univ. Rey Juan Carlos)
José Norberto Mazón (Univ. Alicante)

José Ramón Paramá (Univ. A Coruña)
José Riquelme (Univ. Sevilla)
José Samos (Univ. Granada)
Juan Carlos Trujillo (Univ. Alicante)
Juan de Lara (Univ. Aut. Madrid)
Juan Garbajosa (Univ. Polit. Madrid)
Juan Hernández (Univ. Extremadura)
Juan José Moreno (Univ. Polit. Madrid)
Juan Manuel Murillo (Univ. Extremadura)
Juan Manuel Vara (Univ. Rey Juan Carlos)
Juan Sánchez (Univ. Polit. Valencia)
Luis Iribarne (Univ. Almería)
Mª Esperanza Manso (Univ. Valladolid)
Mª José Escalona (Univ. Sevilla)
Macario Polo (Univ. Castilla-La Mancha)
Manuel Fernández-Bertoa (Univ. Málaga)
Manuel Nuñez (Univ. Comp. de Madrid)
Manuel Resinas (Univ. Sevilla)
Marcela Genero (Univ. Castilla-La Mancha)
María José Aramburu (Univ. Jaume I)
Maribel Sánchez-Segura (U. Carlos III)
Mario Piattini (Univ. Castilla-La Mancha)
Miguel Goulao (Univ. Nova Lisboa)
Miguel R.Luaces (Univ. A Coruña)
Miguel Toro (Univ. Sevilla)
Natalia Juristo (Univ. Polit. Madrid)
Nelly Bencomo
Nieves Brisaboa (Univ. A Coruña)
Orlando Ávila-García (Open Canarias S.L.)
Oscar Díaz (Univ. País Vasco)
Oscar Dieste (Univ. Polit. Madrid)
Oscar Pastor (Univ. Polit. Valencia)
Óscar Pedreira (Univ. A Coruña)
Pablo de la Fuente (Univ. Valladolid)
Patricia Paderewski (Univ. Granada)
Pedro J.Clemente (Univ. Extremadura)
Pedro Pablo Alarcón (Univ. Polit. Madrid)
Pedro Sánchez (Univ. Polit. Cartagena)
Pepe Carsí (Univ. Polit. Valencia)
Rafael Berlanga (Univ. Jaume I)
Rafael Capilla (Univ. Rey Juan Carlos)
Rafael Corchuelo (Univ. Sevilla)
Robert Clarisó (UOC)
Roberto Ruiz (Universidad Pablo Olavide)
Salvador Trujillo (IKERLAN)
Santiago Meliá (Univ. Alicante)
Sergio Segura (Univ. Sevilla)
Sira Vegas (Univ. Polit. Madrid)
Toni Urpí (Univ. Polit. Catalunya)

Valeria De Castro (Univ. Rey Juan Carlos)
Verónica Bollati (Univ. Rey Juan Carlos)
Vicente Luque Centeno (Univ. Carlos III)
Vicente Pelechano (Univ. Polit. Valencia)
Víctor Sánchez (Open Canarias)
Yania Crespo (Univ. Valladolid)

Comité de Organización

Presidente:

Luis Iribarne (Universidad de Almería)

Miembros:

Alfonso Bosch (Universidad de Almería)
Antonio Corral (Universidad de Almería)
Diego Rodríguez (Universidad de Almería)
Elisa Álvarez, Fundación Mediterránea
Javier Criado (Universidad de Almería)
Jesús Almendros (Universidad de Almería)
Jesús Vallecillos (Universidad de Almería)
Joaquín Alonso (Universidad de Almería)
José Andrés Asensio (Universidad de Almería)
José Antonio Piedra (Universidad de Almería)
José Francisco Sobrino (Universidad de Almería)
Juan Francisco Inglés (Universidad Politécnica de Cartagena)
Nicolás Padilla (Universidad de Almería)
Rosa Ayala (Universidad de Almería)
Saturnino Leguizamón (Universidad Tecnológica Nacional, Argentina)

Índice de Contenidos

Sesiones Temáticas

Sesión Temática 1: Bases de Datos, Almacenes de Datos, Minería de Datos, Recuperación de la información.
Coordinadores: Dr. Alfredo Goñi y Dr. José Francisco Aldana

Sesión Temática 2: Ing. Web, Interf. Usuario, Sist. Colaborativos, Computación Ubicua
Coordinadores: Dr. Pascual González y Dr. Juan Carlos Preciado

---

Sesión Temática 3: Apoyo decisión Ing. Software, Metodologías, Experimentación
Coordinadores: Dra. Mercedes Ruiz y Dr. Agustín Yagüe

Sesión Temática 4: Calidad, Pruebas y Requisitos
Coordinadores: Dr. Xavier Franch y Dr. Claudio de la Riva

Sesión Temática 5: *Desarrollo de Software Dirigido por Modelos*
Coordinadores: Dr. Antonio Vallecillo y Dr. José Raul Romero

Sesión Temática 6: Líneas de Producto, Componentes y Arquitecturas Software
Coordinadores: Dr. Carlos Canal y Dr. Silvia Abrahão

Sesión Temática 7: Miscelánea
Coordinadora: Dra. Coral Calero

Charla Invitada

Crossing the Software Education Chasm using Software-as-a-Service and Cloud Computing

Armando Fox

A. Ruíz, L. Iribarne (Eds.): Actas de las "*XVII Jornadas de Ingeniería del Software y Bases de Datos (JISBD'2012)*", Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

Crossing the Software Education Chasm using Software-as-a-Service and Cloud Computing

Prof. Armando Fox
Computer Science Division, University of California, Berkeley
fox@cs.berkeley.edu

Via the remarkable alignment of cloud computing, software as a service (SaaS), and Agile development, the future of software has been revolutionized in a way that also allows us to teach it more effectively. Over the past 3 years we have been reinventing UC Berkeley's undergraduate software engineering course to cross the long-standing chasm between what many academic courses have traditionally offered and the skills that software employers expect in new hires: enhancing legacy code, working with nontechnical customers, and effective testing. In our course, "two-pizza teams" of 4 to 6 students create a prototype application specified by real customers (primarily nonprofit organizations) and deploy it on the public cloud using the Rails framework and Agile techniques. Students employ user stories and behavior-driven design to reach agreement with the customer and test-driven development to reduce mistakes. During four 2-week iterations, they continuously refine the prototype based on customer feedback, experiencing the entire software lifecycle–requirements gathering, testing, development, deployment, and enhancement–multiple times during a 14-week semester. Because of Rails' first-rate tools for testing and code quality, students learn by doing rather than listening, and instructors can concretely measure student progress. We have also successfully repurposed those same tools to support nontrivial machine grading of complete programming assignments, allowing us to scale the on-campus course from 35 to 115 students and offer a Massively Open Online Course (MOOC) to over 50,000 students. Indeed, to support instructors interested in adopting our techniques in their classes, we provide not only an inexpensive textbook and prerecorded video lectures to complement the curriculum, but also a set of questions and programming assignments that includes free autograding. Our experience has been that students love the course because they learn real-world skills while working with a real customer, instructors love it because students actually practice what they learn rather than listening to lecture and then coding the way they always have, and employers love it because students acquire vital skills missing from previous software engineering courses.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 21, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

A. Ruíz, L. Iribarne (Eds.): Actas de las "*XVII Jornadas de Ingeniería del Software y Bases de Datos (JISBD'2012)*", Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

Sesión Temática 1

Bases de Datos, Almacenes de Datos, Minería de Datos, Recuperación de la información.

Coordinadores: *Dr. Alfredo Goñi y Dr. José Francisco Aldana*

Sesión Temática 1: Bases de Datos, Almacenes de Datos, Minería de Datos, Recuperación de la información.
Coordinadores: Dr. Alfredo Goñi y Dr. José Francisco Aldana

Carlos Blanco Bueno, Eduardo Fernandez-Medina and Juan Trujillo. *Modelado Seguro de Consultas OLAP y su Evolución*. (Emergente)

Elisa de Gregorio, et al. *Modelado y Generación Automática de Requisitos de Cuadros de Mando*. (Emergente)

F. J. Fernández Bejarano, *et al*. *MiningDeepWeb: Herramienta para la Extracción de Información en la Web profunda mediante técnicas de minería de datos*. (Herramienta)

Jose-Norberto Mazon, et al. *Open Business Intelligence: uso amigable de tecnicas de inteligencia de negocio sobre datos abiertos*. (Emergente)

David Anton, Alfredo Goñi and Arantza Illarramendi. *Diseño de un sistema de telerehabilitación basado en Kinect.* (Emergente)

Manuel A. Regueiro, et al. *Integración de observaciones medioambientales: Solución inicial y retos futuros*. (Emergente) …

Sebastián Villarroya, et al. *Análisis espacio-temporal en sistemas de bases de datos lógico-funcionales*. (Emergente)

Ismael Navas-Delgado, et al. *Social Pathway Annotation: Extensions of the Systems Biology Metabolic Modelling Assistant*. (Relevante)

Roberto Uribe-Paredes, et al. *Una estructura Metrica Generica para Busquedas por Rango sobre una Plataforma Multi-GPU*. (Regular)

Francisco Claude and Susana Ladra. *Practical Representations for Web and Social Graphs*. (Relevante)

Luis G. Ares, et al. *Reducción de la Complejidad Externa en Búsquedas por Similitud usando Técnicas de Clustering*. (Regular) ..

Angel Luis Garrido, et al. *NASS: A Semantic Annotation Tool for Media*. (Regular)

A. Ruíz, L. Iribarne (Eds.): Actas de las "*XVII Jornadas de Ingeniería del Software y Bases de Datos (JISBD'2012)*", Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

Modelado Seguro de Consultas OLAP y su Evolución

Carlos Blanco[1], Eduardo Fernández-Medina[2], Juan Trujillo[3] y Jan Jurjens[4]

[1]Dep. de Matemáticas, Estadística y Computación. Facultad de Ciencias. Grupo GSyA. Universidad de Cantabria. Av. De los Castros s/n. 39071. Santander. Spain. Carlos.Blanco@unican.es
[2]Dep. de Tecnologías y Sistemas de Información. Escuela Superior de Informática. Grupo GSyA. Universidad de Castilla-La Mancha. Paseo de la Universidad, 4. 13071. Ciudad Real. Spain. Eduardo.Fdezmedina@uclm.es
[3]Dep. de Lenguajes y Sistemas de Información. Facultad de Informática. Grupo LUCENTIA. Universidad de Alicante. San Vicente s/n. 03690. Alicante. Spain. jtrujillo@dlsi.ua.es
[4]Germany TU Dortmund & Fraunhofer ISST. Alemania. jan.jurjens@cs.tu-dortmund.de

Resumen. La seguridad de la información es un aspecto crítico para las organizaciones. Los almacenes de datos manejan información histórica altamente sensible, ya que además de ser el apoyo a la toma de decisiones estratégicas suele incluir datos personales protegidos por ley. Por lo tanto, esta información ha de ser asegurada garantizando que los usuarios finales encargados de la toma de decisiones no accedan ni infieran información no autorizada en sus consultas al almacén mediante aplicaciones OLAP. Este artículo presenta una propuesta para el modelado seguro de consultas OLAP en la que se modelan tanto consultas OLAP sensibles, como su posible evolución mediante la aplicación de operaciones OLAP. Esta propuesta permite por lo tanto establecer la información que le ha de ser proporcionada al usuario en cada momento de su interacción con el almacén, teniendo en cuenta la información que ha ido conociendo previamente para limitar así el riesgo de inferencias.

Palabras Clave: Almacenes de Datos, OLAP, Seguridad, Evolución de Consultas, Modelo de Estados.

1 Introducción

Los almacenes de datos se encargan de almacenar información histórica de negocio de una forma adecuada para facilitar el proceso de análisis y toma de decisiones estratégicas. La propuesta más ampliamente aceptada para su organización es el modelado multidimensional, en el que las medidas a ser analizadas representan la parte central (hechos) y varias dimensiones permiten clasificar estas medidas bajo distintos puntos de vista y niveles de detalle (jerarquías de clasificación) [1, 2].

Los almacenes de datos manejan información altamente sensible: información estratégica de negocio además de información de carácter personal. Por lo tanto, es necesario que se establezcan los mecanismos adecuados para asegurar el almacén frente a accesos no autorizados [3].

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 25-30, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

En el desarrollo de sistemas de información la seguridad ha sido reconocida como un aspecto crítico que ha de ser tenido en cuenta en todas las etapas del proceso de desarrollo [4-6]. Sin embargo, las soluciones aportadas no pueden ser directamente aplicadas a almacenes de datos al no contemplar la estructura y operaciones específicas de los almacenes de datos y OLAP [3].

Las soluciones de seguridad relacionadas con almacenes de datos y OLAP han estado centradas en la incorporación de la seguridad en las etapas finales de desarrollo (modelado físico o implementación). Sin embargo, la incorporación de la seguridad desde etapas tempranas de modelado hace que las decisiones de diseño se tomen teniendo en cuenta estos requisitos y que por lo tanto se vaya generando una solución más robusta y de mayor calidad. Sólo algunos trabajos han propuesto metodologías para el desarrollo seguro de un almacén de datos [7-9] que permiten asociar restricciones de seguridad sobre elementos multidimensionales del almacén, pero que sólo tratan la seguridad desde un punto de vista estático indicando qué información puede observar un determinado usuario según sus privilegios de seguridad.

Sin embargo, el usuario final en su sesión de análisis OLAP sobre el almacén podría realizar una consulta y aplicar una secuencia de operaciones OLAP (dril-down, roll-up) con el objetivo de descubrir o inferir información no autorizada. Con un modelado estático de la seguridad se consigue limitar de antemano cierta información a los usuarios, pero no permite modelar restricciones de seguridad dependientes de la información que el usuario ha ido descubriendo en su interacción con el almacén (evolución de las consultas mediante operaciones OLAP). El control de inferencias es un problema importante de seguridad que puede ser estudiado analizando el paralelismo existente con las bases de datos estadísticas [10, 11], y en el que se han realizado propuestas que tratan de resolverlo perturbando datos o limitando consultas, pero requieren de un gran esfuerzo de computación [12, 13].

Este artículo presenta una propuesta de modelado seguro de consultas OLAP en la que se tienen en cuenta las posibles evoluciones de la consulta mediante operaciones OLAP y la información que va descubriendo el usuario, para en base a ello, indicar qué información se le ha de proporcionar al usuario en cada momento. Estos modelos dinámicos nos permiten complementar una propuesta previa para el modelado conceptual seguro del almacén [14].

Este artículo está organizado de forma que la Sección 2 presenta una visión de la propuesta para el modelado seguro de consultas OLAP y su evolución. La Sección 3 presenta la propuesta mediante un ejemplo de aplicación y finalmente, la Sección 4 las conclusiones y trabajos futuros planteados.

2 Propuesta para el Modelado Seguro de Consultas OLAP y su Evolución

La Figura 1 muestra una visión general de la propuesta de modelado de almacenes de datos y aplicaciones OLAP seguras, considerando aspectos dinámicos de seguridad.

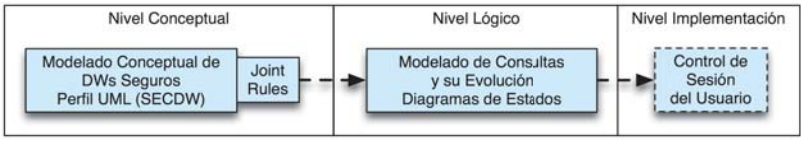


Figura 1. Security Modelling Proposal for OLAP Systems

En primer lugar, se realiza el modelado conceptual del almacén utilizando un perfil UML desarrollado previamente (SECDW) [14, 15]. En este modelo se definen tanto la estructura como las restricciones de seguridad del almacén, que indican los elementos multidimensionales a los que cada usuario puede acceder dependiendo de sus privilegios de seguridad. Estas medidas de seguridad son estáticas, es decir, no tienen en cuenta las sesiones de análisis OLAP que realizan los usuarios, que podrían inferir información sensible con la información que van combinando y descubriendo en sus consultas.

En una segunda etapa, la propuesta complementa el modelado de seguridad estático con un modelado dinámico a nivel lógico, utilizando diagramas de estados. Cada diagrama de estados modela una consulta sensible (identificada en el modelo conceptual por una regla denominada "JointRule") y sus posibles evoluciones al aplicar diferentes operaciones OLAP ("roll-up" y "drill-down"). Para evitar inferencias, en cada posible estado de una consulta sensible, el diseñador indica exactamente la información que ha de mostrarse, basándose para ello en las reglas de seguridad establecidas para el almacén, la información consultada previamente por el usuario y la información a la que puede acceder desde el estado actual utilizando operaciones OLAP.

Finalmente, a nivel de implementación, se ha de controlar la sesión del usuario para evitar que realice una secuencia maliciosa de varias consultas que le permita cruzar datos de forma manual e inferir información sensible.

3 Ejemplo de Aplicación

A continuación se presenta la propuesta mediante su aplicación al modelado de un almacén de ventas que gestiona información de tiendas, productos y clientes.

La Figura 2 muestra el modelo conceptual del almacén, siguiendo la sintaxis del perfil UML SECDW [14]. Para establecer la configuración de seguridad de este ejemplo se ha definido una secuencia de niveles de seguridad: alto secreto (TS), secreto (S), confidencial (C) y sin clasificar (U), siendo el nivel de alto secreto el de mayores privilegios de seguridad.

Estos niveles de seguridad son utilizados para indicar los privilegios necesarios para acceder a cada elemento multidimensional del almacén (representados como valores etiquetados) y para definir reglas de seguridad más complejas como "EnvioAnonimo" que eleva el nivel de seguridad requerido para consultar los datos de productos con envío anónimo de confidencial a secreto.

Tanto la consulta de información de productos como la de clientes, requieren de un nivel de seguridad de confidencial. Sin embargo, identificamos que la consulta de

forma combinada de clientes y productos es más sensible y por lo tanto, plantemos la incorporación de una regla de combinación de información sensible (JointRule) que establece un nivel de seguridad más elevado, el de secreto.

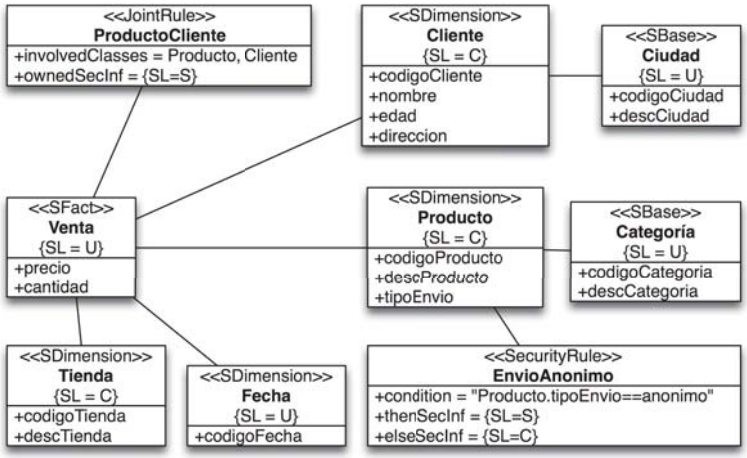


Figura 2. Modelo Conceptual Seguro del Almacén

A continuación, la regla de combinación de información sensible (JointRule) definida es refinada con un modelo de estados que controlará las consultas que combinan productos y clientes, para que su evolución mediante operaciones OLAP ni ponga al descubierto ni permita inferir información no autorizada. La Figura 3 muestra el diagrama de estados correspondiente.

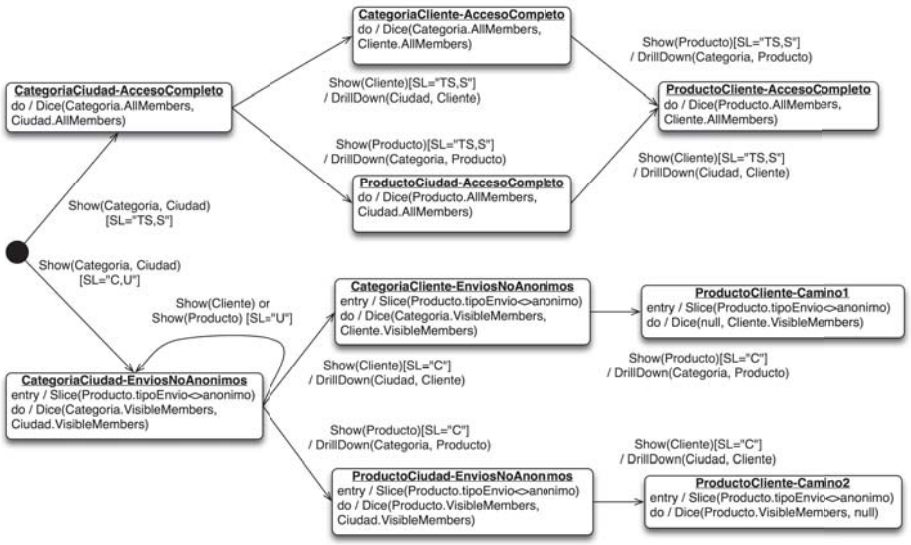


Figura 3. Modelado Seguro de la Consulta "Productos/Clientes"

El diagrama presenta un único punto de entrada al que se conduciran todos los usuarios que soliciten consultar la información de productos y clientes, en el que se les mostrará esta información pero agrupada según el nivel de detalle menos específico, es decir, por categorías de productos y por ciudades. A continuación, los usuarios podrán aplicar operaciones OLAP con objeto de desplazarse por las jerarquías de clasificación implicadas hacia niveles de detalle más o menos específicos (operaciones "dril-down" y "roll-up"). El diagrama de estados contempla todas estas opciones de evolución de la consulta, derivando a los usuarios hacia determinados estados que filtran (operación "slice") y muestran (operación "dice") la información solicitada garantizando que no descubran información no autorizada. La creación de los diferentes estados se realiza teniendo en cuenta las restricciones de seguridad establecidas en el modelo conceptual, los privilegios de seguridad que puede tener el usuario y la información que ha ido descubriendo en su interacción con el almacén (los estados previamente visitados).

Al comenzar la sesión OLAP, la mayoría de los estados a los que puede dirigirse un usuario corresponden a una división realizada de acuerdo a los privilegios de seguridad y las restricciones de seguridad definidas en el modelo conceptual. Sin embargo, según avanza el usuario en su interacción con el almacén las decisiones del diseñador toman más importancia, ya que además de tener en cuenta las restricciones de seguridad ha de tener presente la información que ha descubierto el usuario, para definir y conducir al usuario hacia determinados estados que garanticen que no accede ni infiere información no autorizada. En este sentido, un usuario que desee consultar cierta información puede ser derivado a estados distintos (y por lo tanto observar distinta información) dependiendo de la secuencia de operaciones OLAP que haya realizado.

4 Conclusiones

Los almacenes de datos gestionan información altamente sensible debido a su importancia estratégica para la empresa y a que suele almacenar información privada de carácter personal. Es por lo tanto necesario que la seguridad tenga un papel importante en el proceso de desarrollo del almacén. Las propuestas existentes consideran el modelado estático de la seguridad, sin tener en cuenta la información que el usuario va conociendo en su interacción con el almacén.

Este artículo ha presentado una propuesta para el modelado seguro de consultas OLAP y su evolución. Para ello, dado que el modelado de todas las posibles consultas es algo inabordable, la propuesta se centra en modelar aquellas consultas identificadas como de especial sensibilidad.

Nuestra propuesta representa estas consultas en el modelado conceptual del almacén utilizando reglas de seguridad (JointRule) y a continuación, refina cada una de ellas a nivel lógico utilizando diagramas de estados. Cada usuario dependiendo de sus privilegios de seguridad y de la información que ha consultado previamente, será derivado hacia unos u otros estados al aplicar operaciones OLAP. En el propio estado se indican los filtros necesarios a aplicar para impedir que el usuario pueda consultar o inferir información no autorizada.

Como trabajo futuro identificamos la necesidad de implementar la propuesta proporcionando una herramienta que capture y vigile las consultas y operaciones OLAP del usuario, basándose en los diagramas de estados definidos. Además, debería capturar y analizar la historia de las distintas consultas que realiza el usuario, para poder detectar inferencias indirectas que pudieran obtenerse al cruzar datos de varias consultas no sensibles. Además, planteamos como trabajo futuro la generación automatizada de diagramas de estados base para cada consulta sensible identificada en el modelo conceptual. La automatización completa no sería posible ya que el diseñador tiene un papel activo en su creación pero de este modo se le proporcionaría un punto de partida sobre el cuál poder aplicar esta propuesta de una forma más cómoda.

Agradecimientos. Esta investigación is parte de los proyectos PEGASO/MAGO (TIN2009-13718-C02-01) del Ministerio de Ciencia e Innovación; SERENIDAD (PEII11-0327-7035) de la Consejería de Educación, Ciencia y Cultura de la Junta de Comunidades de Castilla-La Mancha; y MEDUSAS (IDI-20090557) del Centro para el Desarrollo Tecnológico Industrial del Ministerio de Ciencia e Innovación (CDTI).

References

1. Inmon, H., *Building the Data Warehouse*. Third Edition ed2002, USA: John Wiley & Sons.
2. Kimball, R., *The Data Warehouse Toolkit*2002: John Wiley & Sons.
3. Thuraisingham, B., M. Kantarcioglu, and S. Iyer, *Extended RBAC-based design and implementation for a secure data warehouse*. International Journal of Business Intelligence and Data Mining (IJBIDM), 2007. **2**(4): p. 367-382.
4. Fernández-Medina, E., et al., *Model-Driven Development for secure information systems*. Information and Software Technology, 2009. **51**(5): p. 809-814.
5. Jurjens, J., *Secure Systems Development with UML*2004: Springer-Verlag.
6. Mouratidis, H., *Software Engineering for Secure Systems: Industrial and Research Perspectives*2011: IGI Global.
7. Priebe, T. and G. Pernul. *A Pragmatic Approach to Conceptual Modeling of OLAP Security*. in *20th International Conference on Conceptual Modeling (ER 2001)*. 2001. Yokohama, Japan: Springer-Verlag.
8. Fernández-Medina, E., J. Trujillo, and M. Piattini, *Model Driven Multidimensional Modeling of Secure Data Warehouses*. European Journal of Information Systems, 2007. **16**: p. 374-389.
9. Saltor, F., et al., *Building Secure Data Warehouse Schemas from Federated Information Systems*, in *Heterogeneous Inf. Exchange and Organizational Hubs*, H. Bestougeff, Dubois, J.E., Thuraisingham, B, Editor 2002, Kluwer Academic Publisher: Dordrecht, The Netherlands. p. 123-134.
10. Shoshani, A., *OLAP and Statistical Databases: Similarities and Differences*, in *PODS 97*1997: Tucson, AZ.
11. Xie, H., L. Kulik, and E. Tanin, *Privacy-aware collection of aggregate spatial data*. Data \& Knowledge Engineering, 2011. **70**(6): p. 576-595.
12. Zou, L., L. Chen, and M.T. Ozsu, *K-Automorphism: A General Framework For Privacy Preserving Network Publication*. PVLDB, 2009. **2**(1): p. 946-957.
13. Liu, K. and E. Terzi, *Towards identity anonymization on graphs*, in *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada*2008, ACM. p. 93-106.
14. Fernández-Medina, E., et al., *Developing Secure Data Warehouses with a UML extension*. Information Systems, 2007. **32**(6): p. 826-856.
15. Fernández-Medina, E., et al., *Access Control and Audit Model for the Multidimensional Modeling of Data Warehouses*. Decision Support Systems, 2006. **42**: p. 1270-1289.

Modelado y Generación Automática de Requisitos de Cuadros de Mando

Elisa de Gregorio, Alejandro Maté, Hector Llorens, Juan Trujillo

Lucentia Research Group
Department of Software and Computing Systems
University of Alicante
{edg12,amate,hllorens,jtrujillo}@dlsi.ua.es

Resumen La Inteligencia de Negocio (IN) utiliza grandes cantidades de información procedentes de fuentes heterogéneas que tradicionalmente se encuentran integradas en un Almacén de Datos (AD). De forma general, se ha prestado especial atención al proceso de diseño e implementación del AD desde el punto de vista de la información a almacenar. Sin embargo, hasta el momento son pocas las aproximaciones que priorizan las necesidades de explotación de la información por parte de los tomadores de decisión. De esta forma, los tomadores de decisión cuentan con los datos necesarios pero no son capaces de utilizarlos de forma óptima, ni relacionarlos con la estrategia de negocio. En este artículo, proponemos un metamodelo para el diseño de cuadros de mando, que permite a los diseñadores capturar las necesidades de datos de los tomadores de decisión y, posteriormente, obtener la implementación correspondiente en la plataforma de IN objetivo. De esta forma, las necesidades de información y explotación de los usuarios finales guían el proceso de diseño de los cuadros de mando, con el objetivo de aumentar la satisfacción de los usuarios finales.

Key words: Cuadros de mando, Almacenes de datos, Requisitos, Visualización de datos, Modelado conceptual, MDA.

1. Introducción

La Inteligencia de Negocio utiliza grandes cantidades de información para dar soporte al proceso de toma de decisiones. La información utilizada en este proceso se encuentra tradicionalmente almacenada y estructurada en Almacenes de datos (ADs)[10]. Se muestran por medio de diferentes mecanismos de visualización de datos, como los cuadros de mando[7], por permitir la monitorización de múltiples tipos de información, por medio de gráficas o tablas.

Actualmente, en el mercado existe un gran catálogo de soluciones de visualización que permiten la construcción de cuadros de mando, a partir de diferentes fuentes de datos [3,4,17]. Sin embargo, la construcción correcta de los cuadros de mando es una tarea con un alto coste y complejidad, ya que es necesario (i) identificar medidas e Indicadores Clave de Desempeño (KPI, Key Performance

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 31-36, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

Indicators) adecuados para el análisis, (ii) saber exactamente qué objetivos del plan estratégico influyen en los elementos visualizados, y (iii) construir el cuadro de mando dependiendo de la plataforma de inteligencia de negocio, teniendo en cuenta sus particularidades de desarrollo. Por ello, en este artículo se propone un metamodelo para la creación de cuadros de mando, en el que se podrá conceptualizar desde un nivel de abstracción superior la estructura de los cuadros de mando en etapas tempranas del desarrollo.

En trabajos previos [12,13,14], se ha definido un método de desarrollo híbrido de AD en el contexto del marco Model Driven Architecture (MDA) [15]. La automatización del proceso con MDA consigue mejorar el tiempo y esfuerzo de los desarrolladores. En nuestra aproximación, los requisitos se especifican en el Modelo de Computación Independiente (CIM, Computation Independent Model) a partir de un perfil de UML basado en i* [20]. En este nivel estarán modelados los objetivos estratégicos, decisionales e informacionales de la estrategia de negocio. A continuación, se derivan semi-automáticamente, reconciliándose con las fuentes de datos en un modelo híbrido y refinado a través de la capa de Modelo Independientes de la Plataforma (PIM, Platform Independent Model) y la capa del Modelo Específico de la Plataforma (PSM, Platform Specific Model).

La propuesta se apoya y completa en los trabajos previos, permitiendo generar semi-automáticamente el cuadro de mando asociado a cada objetivo de negocio, mediante un conjunto de transformaciones Model-to-Text (M2T) definidas con Acceleo [1]. En el proceso, se vinculan los datos del metamodelo de cuadros de mando, con los datos que corresponden al modelado de requisitos del AD en la capa CIM, influenciado por la estrategia de negocio.

Nuestra aproximación permite, (i) relacionar los elementos del cuadro de mando generado con los objetivos de negocio definidos en los requisitos de ADs, (ii) facilitar el modelado de los KPI, al partir de un análisis de requisitos con los objetivos de la estrategia de negocio ya identificados, y (iii) generar cuadros de mando en la plataforma destino elegida de forma semi-automática. Este metamodelo presenta como ventajas (i) la identificación de errores en la especificación en etapas tempranas del desarrollo, ya que se dispondrá de prototipos que permitan validar los requisitos y además, (ii) permite el cambio de plataforma de inteligencia de negocio al generar los cuadros de mando de forma semi-automática por medio de transformaciones M2T.

El resto del artículo se encuentra estructurado de la siguiente forma. En la Sección 2 se exponen los trabajos relacionados la propuesta. En la Sección 3 se describe la aproximación para la captura, modelado y derivación de requisitos de visualización. La Sección 4 se presenta un ejemplo de aplicación del metamodelo. Finalmente, en la Sección 5, se presentan las conclusiones y los trabajos futuros.

2. Trabajos relacionados

En la literatura existen distintas aproximaciones relacionadas con el modelado de las estrategias de negocio y la creación de ADs. Para el modelado de estrategias del negocio, en [9], los autores proponen alinear las estrategia de nego-

cio con aspectos operacionales de la empresa. Para ello presentan un metamodelo de mapas estratégicos y cuadros de mando balanceados(BSC) con su definición ontológica en OWL [18] y Telos [11] donde se formalizan los conceptos, asociaciones y restricciones. En [8] propone el modelado de mapas estratégicos y BSC usando i* como una técnica de modelado de objetivos. La propuesta descrita en [6] se centra en el modelado de los KPI en el marco MDA, utilizando Semantics Of Business Vocabulary And Rules (SBVR) [16], que ofrece un lenguaje estructurado natural para la definición formal de los KPI. A continuación, se reescriben utilizando un lenguaje matemático basado en XML. Por último definen un vocabulario de KPI para ámbitos específicos extendido por medio de MDA. De las propuestas revisadas, ninguna de ellas incluye la unión de los requisitos con las fuentes de datos, ni la forma en las que se deben visualizar.

En el campo de modelado de AD, se ha trabajado en un framework de modelado i* para la construcción de almacenes, especificando los requisitos en la capa CIM, mediante un perfil de UML. A continuación son derivados y reconciliados con las fuentes de datos [12,13,14].

En el proceso de revisión de la literatura, no se han encontrado ninguna propuesta de modelado y generación de cuadros de mando. Nuestra propuesta presenta una innovación en la visualización de información y al estar integrada en los trabajos previos de requisitos de AD, se facilita el proceso de construcción de cuadros de mando.

3. Metamodelo de creación de cuadros de mando

El metamodelo propuesto se muestra en la Figura 1. Éste metamodelo está compuesto por un elemento *Canvas*, que contiene un conjunto de cuadros de mando. Dentro del elemento *Canvas*, los elementos *Dashboard* representan la visualización que se va a realizar de los datos, e incluyen propiedades tales como el tamaño del cuadro de mando, el título, la descripción con el propósito del análisis de los datos y el autor. Cada uno de los *Dashboard* está compuesto por un conjunto de *GraphicalElement*. Un *GraphicalElement* representa un elemento gráfico incluido en el cuadro de mando, por ejemplo, gráficos de barras. El objeto *GraphicalElement* incluye las propiedades comunes de las diferentes formas de visualización de los datos, como son la posición y el tamaño dentro del cuadro de mando, el título del gráfico y la descripción. También se incluye la consulta MDX[19], que especifica los datos que se van a mostrar y es generada a partir de la información multidimensional del AD. A partir del elemento *GraphicalElement* se especifican diferentes tipos de gráficas, como por ejemplo, gráficas de barras, gráficas lineales o gráficas circulares. Cada uno de estos elementos posee sus atributos característicos propios de visualización. En nuestro metamodelo, los KPI se podrán modelar por medio del objeto *BulletChart* que hereda del elemento *GraphicalElement*, y posee los campos para los valores objetivo (*targetValue*), umbral (*thresholdValue*) y peor (*worstValue*) del indicador en cuestión.

Utilizando el metamodelo presentado en la Figura 1 se capturan los requisitos de explotación de la información mediante cuadros de mando para, posterior-

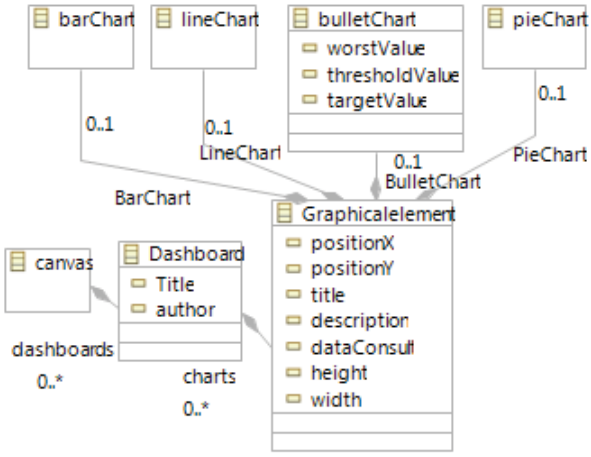


Figura 1: Metamodelo de requisitos de creación de cuadro de mando.

mente, ser derivados de forma semi-automática en un prototipo inicial del cuadro de mando en la plataforma de inteligencia de negocio objetivo.

Por un lado, esta aproximación permite que se enlacen los distintos elementos presentes en los cuadros de mando con los objetivos incluidos en la estrategia del negocio, modelados en los requisitos del AD. De esta forma, se obtiene una visión clara de cómo están siendo analizados cada uno de los objetivos e indicadores de la empresa. Por otro lado, el modelo de cuadro de mando se deriva mediante transformaciones M2T[1] diseñadas con Acceleo, obteniendo rápidamente un prototipo inicial que puede ser validado por el usuario y permitiendo simplificar el cambio de plataforma de inteligencia de negocio, ya que basta con diseñar el código de derivación a la nueva solución a partir del modelo.

Por lo tanto, mediante nuestra aproximación, los requisitos especificados por el usuario se tienen en cuenta y son respetados, permitiendo optimizar la explotación de la información por parte de los tomadores de decisión.

4. Ejemplo de Aplicación

En esta sección se presenta un ejemplo de aplicación de nuestra aproximación, frente a la implementación manual de cuadros de mando.

Para probar nuestra propuesta, se han implementado en la plataforma Eclipse [2] con Acceleo las transformaciones M2T necesarias para generar un cuadro de mando simple, que se puede observar en la parte izquierda de la Figura 2. Como ejemplo, a la derecha en la Figura 2 se ha construido un prototipo de cuadro de mando con datos obtenidos a través de la iniciativa Open Data [5] que contienen información referente a indicadores de sostenibilidad mundiales. El cuadro de

[1] Por cuestiones de espacio no se han incluido las transformaciones.

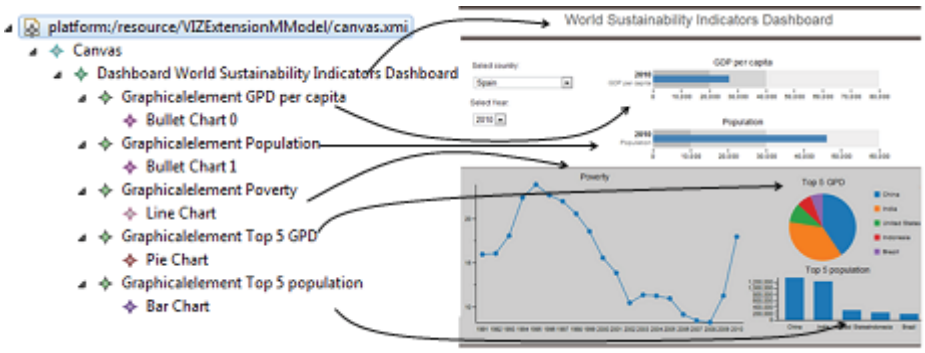


Figura 2: Correspondencia de metamodelo y prototipo de cuadro de mando.

mando se ha modelado en la plataforma Open Source de Pentaho [3]. En la Figura se puede ver como contamos con un elemento *Dashboard* cuyo título es "World Sustainability Indicators Dashboard" y que contiene 5 *GraphicalElement*, correspondientes a cada uno de los gráficos utilizados en el cuadro de mando. Estos elementos contienen la información básica referente a cada uno de los elementos mostrados, y pueden ser posteriormente derivados en la plataforma destino, obteniendo así un prototipo a partir del cual refinar el cuadro de mando, cambiando su estilo o añadiendo, por ejemplo, los elementos de selección deseados. Entre las mejoras esperadas en el proceso de desarrollo se incluye una mayor satisfacción del usuario, ya que puede definir con mayor claridad el cuadro de mando deseado, así como una mayor agilidad para la adaptación de los cuadros de mando ante cambios en la estrategia del negocio y una reducción en el tiempo de desarrollo.

5. Conclusiones y trabajos futuros

En este artículo se ha presentado una propuesta de modelado para la captura y derivación de cuadros de mando en ADs, y su posterior derivación semi-automática en la plataforma de inteligencia de negocio objetivo. Los beneficios de nuestra aproximación son (i) la capacidad de generar una primera versión de los cuadros de mando, basándonos en los requisitos de visualización capturados mediante el metamodelo de una forma rápida y clara, (ii) la incorporación transparente al proceso de los elementos de la estrategia de negocio, ya que se permite enlazar a cada objetivo de la estrategia de negocio con los cuadros de mando correspondientes, y (iii) la disminución del coste y el esfuerzo del desarrollo, al permitir la generación semi-automática de prototipos, disminuyendo la probabilidad de errores de implementación y permitiendo una validación temprana de los cuadros de mando.

En este artículo se han presentado los elementos iniciales del metamodelo para la creación de cuadros de mando. Como trabajo futuro, a corto plazo se

completaran transformaciones M2T para más herramientas de inteligencia de negocio; a medio plazo, se incluirán nuevos elementos gráficos para mejorar la visualización actual como por ejemplo: cubos OLAP y además, se abordará la visualización del alineamiento de los indicadores con la estrategia de negocio.

Agradecimientos. Este trabajo ha sido financiado por el proyectos MESOLAP (TIN2010-14860) del MICINN. Elisa de Gregorio está financiada por el MICINN mediante una beca FPI (BES-2011-043577).

Referencias

1. Acceleo, http://www.eclipse.org/acceleo/(25/04/2012)
2. Eclipse, http://www.eclipse.org/(25/04/2012)
3. Pentaho, http://www.pentaho.com/(25/04/2012)
4. Qlikview, http://www.qlikview.com/(25/04/2012)
5. Un global pulse sustainable development indicators, http://www.visualizing.org/datasets/un-global-pulse-sustainable-development-indicators(25/04/2012)
6. Caputo, E., Corallo, A., Damiani, E., Passiante, G.: KPI modeling in MDA perspective. In: OTM Workshops. pp. 384–393 (2010)
7. Eckerson, W.: Performance dashboards: measuring, monitoring, and managing your business. Wiley (2010)
8. Giannoulis, C., Petit, M., Zdravkovic, J.: Modeling business strategy: A metamodel of strategy maps and balance scorecards. In: RCIS. pp. 1–6 (2011)
9. Giannoulis, C., Zdravkovic, J.: Modeling strategy maps and balanced scorecards using *i*. In: iStar. pp. 90–95 (2011)
10. Kimball, R.: The data warehouse toolkit. Wiley-India (2009)
11. Magnan, F., Paquette, G.: TELOS: An ontology driven elearning OS. In: Workshops held at the Fourth International Conference on Adaptative Hypermedia and Adaptative Web-Based Systems. pp. 131–139 (2006)
12. Mazón, J.N., Pardillo, J., Trujillo, J.: A model-driven goal-oriented requirement engineering approach for data warehouses. In: ER Workshops. pp. 255–264 (2007)
13. Mazón, J.N., Trujillo, J.: An MDA approach for the development of data warehouses. Decision Support Systems 45(1), 41–58 (2008)
14. Mazón, J.N., Trujillo, J., Lechtenbörger, J.: Reconciling requirement-driven data warehouses with data sources via multidimensional normal forms. Data Knowl. Eng. 63(3), 725–751 (2007)
15. OMG: A Proposal for an MDA Foundation Model (2005)
16. OMG: Semantics of business vocabulary and rules (SBVR) (Jan 2008), http://www.omg.org/spec/SBVR/(25/04/2012)
17. SAP: Business objects, http://www.sap.com/solutions/sapbusinessobjects/large/business-intelligence/index.epx(25/04/2012)
18. W3C: OWL, web ontology language, http://www.w3.org/2004/OWL/(25/04/2012)
19. Whitehorn, M., Zare, R., Pasumansky, M.: Fast track to MDX. Springer-Verlag New York Inc (2006)
20. Yu, E.S.K.: Modelling strategic relationships for process reengineering. Ph.D. thesis, Toronto, Ont., Canada, Canada (1995)

MiningDeepWeb: Herramienta para la Extracción de Información en la Web profunda mediante técnicas de minería de datos

Fco. Javier Fdez, Pedro J. Abad, José L. Álvarez, José L. Arjona
Departamento de Tecnologías de la información. Universidad de Huelva
E.P.S. La Rábida, Ctra. Huelva-La Rábida, 21071 Huelva

{javier.fernandez,abadhe,alvarez,jose.arjona}@dti.uhu.es

1 Introducción

El World Wide Web (WWW) fue creado para ofrecer información a las personas, mediante páginas con información visual. Sin embargo, la evolución de la web ha sido tal que para tratar el volumen de datos generado, hoy en día, se requieren técnicas automáticas que permitan la extracción de datos para poder alcanzar un mayor beneficio de la información disponible

La Extracción de información es una modalidad del campo de la recuperación de la información cuyo objetivo es extraer automáticamente información estructurada o semiestructurada desde documentos digitales [1]. Su aplicación sobre documentos HTML, en Internet, ofrece una alternativa a los Servicios Web y la Web Semántica, para el procesamiento automático de la información cuando se carece de éstos.

En este artículo presentamos la herramienta MiningDeepWeb (Mining Information Extraction on Deep Web) que utiliza técnicas de aprendizaje supervisado para la extracción de información sobre páginas Web, a partir de un conjunto de datos que caracterizan la representación visual o renderización de los documentos HTML en un navegador.

El proceso de extracción con MiningDeepWeb requiere de los siguientes pasos: (i) anotación de los elementos de interés en los documentos HTML de un sitio Web; (ii) Generación del conjunto de datos con las características de cada uno de los elementos extraidos; (iii) preprocesado del conjunto de datos para generar el conjunto de entrenamiento, (iv) inducción del modelo (conjunto de reglas) y (v) extracción de la información.

2 Anotación de los elementos de interés y Generación del conjunto de datos.

MiningDeepWeb es una herramienta de extracción de información basada en técnicas de aprendizaje supervisado, que como primer paso requiere de la anotación de páginas webs del sitio en estudio. El proceso de marcado consiste en seleccionar los elementos de interés de cada una de las páginas, estableciendo para cada uno de ellos la clase

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 37-40, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

a la que pertenecen (ver figura 1). Estas clases pueden estar definidas en una ontología previamente cargada o creada ex profeso.
Fruto del proceso de anotación se generará una base de datos que contendrá las características de cada elemento de interés así como la clase a la que pertenece. Además el proceso de marcado va definiendo la zona de interés o área de la página donde se localizan los elementos de interés. Cada sitio web tiene una única zona de interés.

Fig. 1. Proceso de anotación de un elemento de una página, asociándolo a una clase

Las características que se estudian o se recogen de los elementos de interés en las páginas webs, podemos clasificarlos en tres tipos [3]:

- Posición: Coordenadas X e Y, longitud, altura, distancia y pendiente.
- Estilo. Color, fuente, bordes, fondo, etc.
- Contenido/Contexto: Texto, densidad numérica, de dígito, texto anterior, posterior, etc.

El resto de elementos incluidos en la zona de interés, pero que no hayan sido elementos seleccionados previamente, serán clasificados automáticamente como "No Interés", como puede verse en la figura 2.

Fig. 2. Zona de Interés y anotación automática de elementos de "No Interés"

Para aplicar técnicas de minería de datos que permitan analizar la información de las páginas obtenidas del sitio web, MinindDeepWeb ofrece la posibilidad de tratar los datos mediante la herramienta Weka [2].

3 Preprocesado, Inducción y Extracción de Información

Previo a la aplicación de minería de datos, con el objetivo de reducir el conjunto de datos y características a estudiar, MiningDeepWeb permite aplicar diferentes algoritmos de selección de características y evaluadores.

Así mismo, se permite la generación del conjunto de datos de forma balanceada.

El siguiente paso es la obtención de un modelo de conocimiento a partir de los conjuntos de datos, modelo que se obtendrá a partir de la aplicación de algoritmos de clasificación.

Dicho modelo será almacenado para su aplicación posterior, para la extracción de información, en nuevas páginas del sitio para el que se ha obtenido el modelo.

Una vez inferido el modelo para el *Website*, la herramienta permite obtener los distintos elementos de interés del conjunto de páginas que se deseen analizar, devolviendo dichos resultados almacenados y catalogados en formato de base de datos.

Para ello aplica el modelo obtenido al conjunto de datos de cada página a analizar, guardando en una base de datos los elementos de interés, catalogándolos acorde a la clase a la que pertenezca.

Además la herramienta proporciona información en pantalla del proceso de extracción- reconocimiento de cada página tratada, proceso que puede realizarse de forma interactiva o automatizada.

Fig. 3. Extracción de información aplicando modelo.

4 Conclusiones.

Los resultados obtenidos, con la metodología expuesta en el modo de trabajo de la Herramienta MiningDeepWeb, son buenos y reafirman la eficiencia de la aplicación. Sirva de referencia los buenos resultados de aplicación sobre los sitios webs principales de ventas de libros referenciados por Alexa, uno de los más antiguos y expandidos monitores de internet, como se recoge en [3] y que se muestra de forma gráfica en la figura 4.

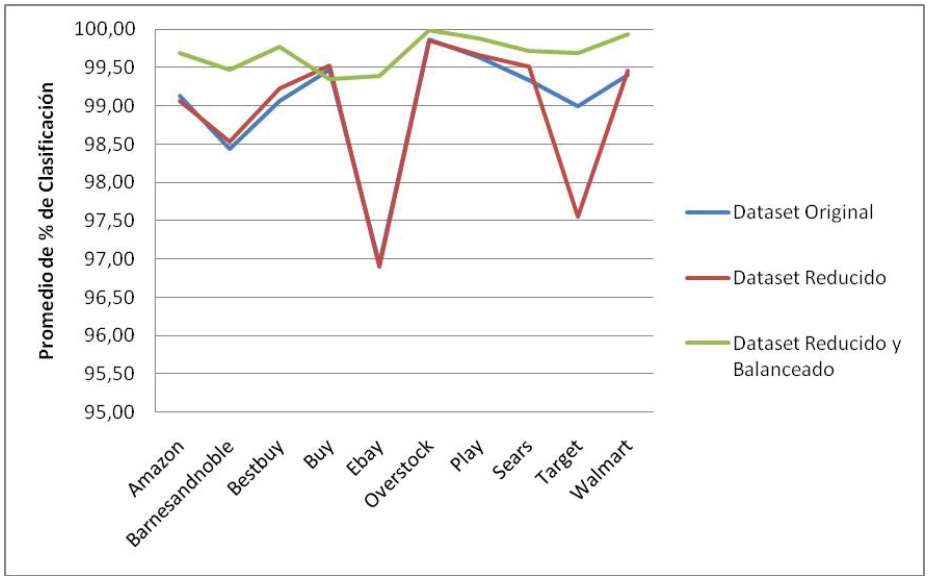


Fig. 4. Resultados obtenidos en webs de ventas de libros

5 Referencias

1. C.-H. Chang, M. Kayed, M. R. Girgis, K. F. Shaalan: A Survey of Web Information Extraction Systems. IEEE Trans. Knowl. Data Eng. 18(10): 1411-1428 (2006)
2. M. Hall, E. Frank, et al. (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1
3. F. J. Fernández et al. Mining Web Pages Using Features of Rendering HTML Elements in the Web Browser. Trends in Practical Applications of Agents and Multiagent Systems Advances in Intelligent and Soft Computing, 2011, Volume 90/2011, pp 161-168

Open Business Intelligence: uso amigable de técnicas de inteligencia de negocio sobre datos abiertos

Jose-Norberto Mazón[1], Jose Jacobo Zubcoff[2], Irene Garrigós[1], Roberto Espinosa[3], and Rolando Rodríguez[3]

[1] Grupo de investigación WaKe, Dept. de Lenguajes y Sistemas Informáticos
Universidad de Alicante
{jnmazon,igarrigos}@dlsi.ua.es
[2] Grupo de investigación WaKe, Dept. de Ciencias del Mar y Biología Aplicada
Universidad de Alicante
jose.zubcoff@ua.es
[3] Grupo de investigación WaKe, Dept. de Informática
Universidad de Matanzas "Camilo Cienfuegos", Cuba
{roberto.espinosa,rolando.rodriguez}@umcc.cu

Resumen La ciudadanía reclama un comportamiento cada vez más transparente de las instituciones públicas. Esta transparencia implica la necesidad de tener disponibles datos cuyo análisis permita a la ciudadanía una participación más activa con el objetivo de proveer el mayor beneficio a la sociedad en su conjunto. Por lo tanto, los datos públicos deben estar disponibles libremente para su reutilización y redistribución, es decir, deben ser datos abiertos. Estos datos abiertos, normalmente se comparten como datos crudos ("raw data") lo que dificulta al ciudadano medio su análisis para obtener conocimiento útil. Se necesita, entonces, mecanismos que permitan a los ciudadanos comprender y analizar los datos públicos de una manera amigable y sencilla. Con esta finalidad, en este artículo se presenta el concepto de Open Business Intelligence (OpenBI). OpenBI facilita a los usuarios no expertos (i) el análisis y visualización de datos abiertos generando de manera sencilla informes, análisis multidimensional, cuadros de mando o minería de datos, (ii) la reutilización, como datos abiertos, de la información y conocimiento adquirido.

1. Introducción

Los ciudadanos esperan un comportamiento cada vez más transparente de las instituciones públicas. Esta transparencia implica la disponibilidad de los datos públicos con el objetivo de que la ciudadanía tome decisiones informadas, posibilitando una participación democrática más activa y maximizando así el beneficio de toda la sociedad. Por lo tanto, se debe poder acceder libremente a los datos públicos con el fin de que puedan ser reutilizados y redistribuidos de manera fácil por cualquier persona, es decir, los datos públicos deben ser

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 41-46, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

abiertos [12,5]. Cabe resaltar que, para que los datos puedan ser ofrecidos de manera "abierta", estos deben poseer una licencia adecuada y estar en formatos abiertos, fácilmente accesibles como CSV, XML o tablas HTML [11].

El movimiento de datos abiertos ("open data") fue impulsado por Tim Berners-Lee en su llamamiento a compartir datos libremente mediante el uso de la Web para el beneficio de toda la comunidad [1], así como en algunos resultados interesantes obtenidos un tiempo después de dicho llamamiento [2]. Sin embargo, no sólo es importante contar con datos abiertos para potenciar la accesibilidad de los datos públicos. Se debe tener en cuenta que la gran mayoría de la ciudadanía es totalmente inexperta en la gestión y análisis de datos, por lo que no sabe cómo conseguir información útil de estas fuentes de datos abiertos. Se puede dar la paradoja de que, siendo los datos públicos accesibles, la información y conocimiento que se puede extraer a partir de ellos no lo sea, y por tanto los datos no serán de ninguna utilidad a la ciudadanía.

Por consiguiente, se requieren mecanismos que permitan a la ciudadanía analizar datos públicos de manera amigable con el fin de poder comprenderlos en su totalidad, reutilizarlos y usarlos en su vida cotidiana (por ejemplo, para conocer el impacto real de la subida de las tasas universitarias o del precio del carburante). De esta manera, además, la ciudadanía podrá entender mejor el funcionamiento de las instituciones públicas, percibiéndose estas como más transparentes y cercanas.

Uno de estos mecanismos se introduce en este trabajo emergente: Open Business Intelligence (OpenBI). OpenBI se define como un conjunto de técnicas que permita a los usuarios inexpertos la integración de diferentes fuentes de datos abiertos con el fin de facilitar (i) el análisis y visualización de datos abiertos generando de manera sencilla informes, análisis multidimensional, cuadros de mando o minería de datos, (ii) la reutilización, como datos abiertos, de la información y conocimiento adquirido. El concepto de OpenBI requiere el desarrollo de propuestas para que los usuarios inexpertos puedan obtener y compartir conocimiento confiable y útil de las fuentes de datos abiertas disponibles.

En este trabajo se presentan las primeras ideas acerca del concepto OpenBI, específicamente se presenta una propuesta que resalta la importancia de los criterios de calidad de datos para guiar a usuarios inexpertos en la extracción y compartición de conocimiento útil a partir de datos abiertos. En concreto nos enfocamos en el uso de técnicas de minería de datos.

El resto de este artículo se estructura de la siguiente manera. En la sección 2 se describen brevemente algunos trabajos relacionados con la calidad y minería de datos. La sección 3 define nuestra propuesta OpenBI para ayudar a los usuarios no expertos en la tarea de aplicar técnicas de minería de datos. Las conclusiones y el trabajo futuro se describen en la sección 4.

2. Trabajo relacionado

Uno de los retos más interesantes del OpenBI es proveer a usuarios no expertos de guías para el preprocesado y posterior aplicación de técnicas de análisis de

datos. Por ejemplo, en el caso de la minería de datos [7], realizar un preprocesado de los datos para conocer y asegurar su calidad es vital para aplicar algoritmos de minería de datos que permitan generar conocimiento pertinente [9]. De hecho, la minería de datos es un paso más en un proceso global conocido como "knowledge discovery in databases" (KDD) (ver Fig. 1). En este proceso, el tratamiento de la calidad de los datos requiere, en muchas ocasiones, más esfuerzo que la propia tarea de minería [8]. Los usuarios inexpertos, por tanto, necesitan de mecanismos para hacer la minería de datos amigable, por ejemplo mediante la automatización del preprocesado o de la adición de interactividad con el usuario en el proceso KDD para considerar aspectos de calidad de datos [9].

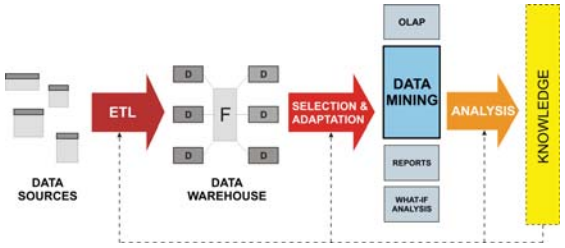


Figura 1. El proceso KDD: extracción de conocimiento a partir de datos

De igual manera, existen propuestas que consideran la necesidad de tener en cuenta criterios de calidad de datos en otras técnicas de inteligencia de negocio. Por ejemplo, en [4], se involucra a los usuarios en la gestión de la calidad de datos relacionada con la generación de informes en aplicaciones de inteligencia de negocio.

3. Una propuesta OpenBI para minería de datos

Esta sección ofrece una visión general de nuestra propuesta para guiar a los usuarios no expertos en el uso de técnicas de minería de datos. Mediante esta propuesta, la ciudadanía podrá analizar y visualizar datos abiertos dentro del escenario OpenBI.

La propuesta consiste en dos etapas principales (tal y como se observa en la Fig. 2): (i) creación de una base de conocimiento a partir de una serie de experimentos para analizar diferentes criterios de calidad de datos en fuentes de datos abiertas y cómo estos criterios de calidad de datos afectan a los resultados de la aplicación de técnicas de minería de datos; y (ii) uso de esta base de conocimiento para aconsejar a los usuarios no expertos sobre la selección de las técnicas más adecuadas para el análisis de los datos (por ejemplo, qué algoritmo de minería de datos debe aplicarse).

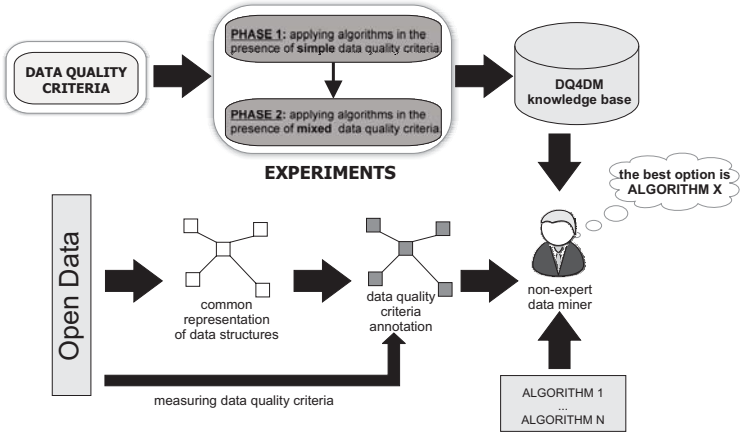


Figura 2. Visión general de nuestra propuesta de OpenBI

3.1. Experimentos para obtener una base de conocimiento

OpenBI implica la realización de un conjunto de experimentos con el fin de evaluar el efecto de varios criterios de calidad de datos en el comportamiento de diferentes técnicas de minería de datos. El objetivo de estos experimentos es la creación de una base de conocimiento que pueda usarse para guiar a los usuarios inexpertos en la obtención de conocimiento a partir de los datos abiertos. En nuestro trabajo previo [6], se ha empezado a considerar el diseño y el desarrollo de estos experimentos.

En este artículo se considera la definición de calidad de datos como "adecuación al uso" [13] lo que implica que los datos deben cumplir con ciertos requisitos con el fin de que puedan usarse para una tarea específica en un contexto concreto. En minería de datos, esto significa que las fuentes de datos deben ser útiles para extraer conocimiento a partir de ellas cuando se analizan usando una u otra técnica. Por ejemplo, si algunos atributos se seleccionan como entrada para un algoritmo de clasificación (estando fuertemente correlacionados), el patrón de conocimiento resultante, aunque correcto, no será útil. En consecuencia, los criterios de calidad de datos que afecten a las técnicas de minería (como la correlación entre atributos de entrada) deben considerarse para evitar usar una técnica de minería que haga a los usuarios inexpertos descubrir conocimiento superfluo, contradictorio o falso. Existen diversos criterios de calidad que permiten determinar en qué medida se pueden usar los datos para la aplicación de técnicas de minería de datos [3,6].

Nuestro método para la obtención de la base de conocimiento se basa en los siguientes pasos:

1. Datos de entrada: son conjuntos de datos conocidos (tanto su calidad de datos como los resultados obtenidos al aplicar técnicas de minería de datos). Además de los datos de entrada, se debe considerar como datos de entrada

el perfil de usuario (es decir, qué se requiere analizar) como entrada, con el fin de conocer qué criterios de calidad de datos evaluar.
2. Preparación de los datos: en esta etapa se crean conjuntos de datos de prueba según el perfil de usuario. En concreto, se crean dos tipos de conjuntos de prueba: el primero considera cada criterio de calidad de datos de manera individual y el segundo combina varios criterios de calidad de datos.
3. Aplicación de experimentos: los experimentos se aplican según el tipo de técnica de minería de datos seleccionada por los usuarios en el primer paso y sobre los datos preparados en el segundo paso.
4. Base de conocimiento: los resultados de los experimentos se incluyen en una base de conocimiento. Esta base de conocimiento podría representarse mediante tripletas RDF (Resource Description Framework)[4] y, más concretamente, mediante la creación de un nuevo vocabulario.

Una vez que se obtiene la base de conocimiento, esta puede usarse en un escenario OpenBI para que un usuario no experto pueda usar, de manera fiable, técnicas de minería de datos sin preocuparse de la calidad de los datos.

3.2. Minería de datos abiertos

Esta sección describe nuestra propuesta para guiar al usuario en la aplicación de técnicas de minería de datos sobre datos abiertos. Consiste en dos etapas: (i) creación de una representación común de los datos abiertos y (ii) medición de los criterios de calidad de datos sobre la representación común (previa definición de cómo calcular estos criterios en esta representación común). Tal y como se muestra en la Fig. 2, nuestra propuesta tiene como finalidad guiar al usuario no experto en el análisis de los datos abiertos de la manera más apropiada.

Creación de una representación común de datos. Con el fin de crear una representación común de datos independiente de la tecnología subyacente, uno de los candidatos es *Common Warehouse Metamodel* (CWM) [10]. CWM es un estándar para representar estructuras de datos e información relacionada. Por lo tanto, las fuentes de datos pueden extraerse en un modelo.

Anotación de criterios de calidad de datos. Una vez que la representación común de los datos abiertos está contenida en un modelo, los criterios de calidad deben medirse y añadirse a este modelo. Este modelo anotado se usa para guiar al usuario no experto en el análisis de los datos a través de técnicas de minería de datos.

4. Conclusiones

Este artículo presenta el concepto de Open Business Intelligence (OpenBI). OpenBI facilita a los usuarios no expertos (i) el análisis y visualización de

[4] `http://www.w3.org/RDF/`

datos abiertos generando de manera sencilla informes, análisis multidimensional, cuadros de mando o minería de datos, (ii) la reutilización, como datos abiertos, de la información y conocimiento adquirido. Uno de los retos más interesantes del OpenBI es el uso de técnicas de minería de datos por usuarios inexpertos que necesitan de guías para el preprocesado y aplicación de algoritmos de minería debido a la baja calidad de los datos abiertos y a la complejidad intrínseca a los mecanismos de minería. En este artículo, definimos un marco general que tenga en cuenta la calidad de los datos con el fin de guiar al usuario en obtener conocimiento útil a partir de técnicas de minería de datos.

Este trabajo es un primer paso hacia la consideración de manera sistemática y estructurada de criterios de calidad de datos para apoyar a usuarios inexpertos en la aplicación de técnicas de inteligencia de negocio sobe datos abiertos.

Agradecimientos. Este trabajo ha sido financiado por los proyectos IN.MIND de la Universidad de Alicante y MANTRA (GV/2011/035) de la Generalitat Valenciana.

Referencias

1. Berners-Lee, T.: On the next Web. `http://www.ted.com/talks/tim_berners_lee_on_the_next_web.html` (2009)
2. Berners-Lee, T.: The year open data went worldwide. `http://www.ted.com/talks/tim_berners_lee_the_year_open_data_went_worldwide.html` (2010)
3. Berti-Equille, L.: Measuring and modelling data quality for quality-awareness in data mining. In: Guillet, F., Hamilton, H.J. (eds.) Quality Measures in Data Mining, Studies in Computational Intelligence, vol. 43, pp. 101–126. Springer (2007)
4. Daniel, F., Casati, F., Palpanas, T., Chayka, O., Cappiello, C.: Enabling better decisions through quality-aware reports in business intelligence applications. In: Neely, M.P., Pipino, L., Slone, J.P. (eds.) ICIQ. pp. 310–324. MIT (2008)
5. Datos.gob.es: Catálogo de Información Pública de la Administración General del Estado. `http://datos.gob.es`
6. Espinosa, R., Zubcoff, J.J., Mazón, J.N.: A set of experiments to consider data quality criteria in classification techniques for data mining. In: Murgante, B., Gervasi, O., Iglesias, A., Taniar, D., Apduhan, B.O. (eds.) ICCSA (2). Lecture Notes in Computer Science, vol. 6783, pp. 680–694. Springer (2011)
7. Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P.: Knowledge discovery and data mining: Towards a unifying framework. In: KDD. pp. 82–88 (1996)
8. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann (2000)
9. Kriegel, H.P., Borgwardt, K.M., Kröger, P., Pryakhin, A., Schubert, M., Zimek, A.: Future trends in data mining. Data Min. Knowl. Discov. 15(1), 87–97 (2007)
10. Object Management Group: Common Warehouse Metamodel Specification 1.1. `http://www.omg.org/cgi-bin/doc?formal/03-03-02`
11. Open Data Commons: Legal tools for Open Data. `http://opendatacommons.org/`
12. Proyecto Aporta: Reutilización de la información del sector público. `http://www.aporta.es`
13. Strong, D.M., Lee, Y.W., Wang, R.Y.: 10 potholes in the road to information quality. IEEE Computer 30(8), 38–46 (1997)

Diseño de un sistema de telerehabilitación basado en Kinect

David Antón[1], Alfredo Goñi[2], Arantza Illarramendi[3]

Departamento de Lenguajes y Sistemas Informáticos. UPV/EHU.
Paseo Manuel de Lardizabal, 1
Donostia, 20018 - San Sebastián

{[1]danton004, [2]alfredo, [3]a.illarramendi}@ehu.es

Abstract. El envejecimiento de la población y la mayor supervivencia frente a enfermedades que dejan secuelas físicas son aspectos que suponen un reto en el contexto de una gestión sanitaria eficiente. Es por ello que en la actualidad se están desarrollando sistemas de telerehabilitación que permiten el seguimiento y apoyo de sesiones de fisioterapia realizadas en el hogar, que sin disminuir la calidad del servicio proporcionado, permiten ahorrar costos sanitarios. Nuestra propuesta es crear un sistema de telerehabilitación basado en Kinect, que a diferencia de otras tecnologías existentes, permite que la persona que debe realizar la rehabilitación no tenga que llevar ningún tipo de sensor en el cuerpo y donde la interacción sea únicamente gestual. Así nuestra propuesta va encaminada por un lado, hacia un reconocimiento local de movimientos y gestos con el fin de evaluar automáticamente los ejercicios terapéuticos realizados por la persona y por otro lado, hacia una comunicación externa con el fisioterapeuta. Otras funcionalidades novedosas del sistema propuesto son: 1. la posibilidad de incorporar nuevos ejercicios de manera simple y acorde al esquema que se sigue en las terapias convencionales, 2. permitir la modificación automática de la terapia o sugerir cambios en función de los resultados de la persona. Tanto en la evaluación de ejercicios como en el análisis de datos se hace uso de técnicas de reconocimiento de gestos, movimientos y análisis estadístico.

Keywords: Telerehabilitación, Kinect, Seguimiento de movimiento, Terapia virtual, Reconocimiento de actividades

1 Introducción

El envejecimiento de la población y la mayor tasa de supervivencia frente a enfermedades que pueden dejar secuelas físicas suponen un reto para una gestión sanitaria eficiente. Por esta razón es necesario implantar nuevos sistemas que permitan la rehabilitación de las personas en el hogar y que a la vez sean efectivos y fáciles de usar para las personas. Los sistemas de telerehabilitación pueden mejorar la calidad de vida de estas personas a la vez que suponen un gran ahorro para los servicios sanitarios.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 47-52, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

En general, un sistema de telerehabilitación permite el seguimiento y apoyo de sesiones de fisioterapia de diferentes colectivos: personas mayores, discapacitados y enfermos, facilitando el contacto con el personal asistencial y mejorando así su calidad de vida. Existen diversos estudios que indican la utilidad terapéutica que tiene el trabajar con sistemas de telerehabilitación. Pruebas realizadas basadas en la interacción virtual han evidenciado que pueden ser tan eficaces como los tratamientos tradicionales y pueden aportar además otras ventajas para el usuario y el fisioterapeuta. [8,9]. Además un factor importante a tener en cuenta es el carácter motivador que pueden tener estos sistemas. Es relativamente frecuente el abandono de las sesiones de rehabilitación clásica por aburrimiento o desinterés. Usar sistemas de telerehabilitación con captura de movimientos puede incrementar la intensidad de la rehabilitación y también la diversión del usuario [3, 8].

Un sistema de telerehabilitación básico tiene como mínimo una cámara que permite al fisioterapeuta ver al usuario. Los sistemas más complejos incluyen sensores que pueden registrar los movimientos del usuario. Existe una gran variedad de métodos de interacción en los que se monitoriza el movimiento de una persona. Estos métodos se pueden dividir en función del tipo de sensor utilizado en tres grupos principales: seguimiento asistido por robot, seguimiento no visual y seguimiento visual [10]. El objetivo de todos ellos es obtener datos en tiempo real de los cambios de posición de una persona y de la partes de su cuerpo.

Nosotros hemos considerado usar Kinect, un sistema de interacción natural desarrollado por Microsoft, como dispositivo de captura de movimientos. Concretamente la versión lanzada en febrero de 2012, optimizada para funcionar en Windows [5]. Kinect se clasifica como un sistema visual sin marcadores que permite a los usuarios controlar e interactuar con las aplicaciones mediante una interfaz que reconoce gestos, comandos de voz y objetos, sin necesidad de tener contacto físico. Consiste en una cámara de vídeo, una cámara de profundidad basada en infrarrojos y una serie de cuatro micrófonos. Los datos obtenidos permiten visualizar la escena en tres dimensiones y proporcionan información sobre la posición y las articulaciones del usuario. En comparación con otros sistemas en los que el usuario tiene que llevar sensores en el cuerpo, los sistemas sin marcadores como Kinect, son más cómodos y no sufren problemas de oclusión del marcador. Esta tecnología aplicada al campo de la telerehabilitación permite crear sistemas que, mediante el reconocimiento de movimientos y gestos, puedan evaluar automáticamente ejercicios terapéuticos realizados por el usuario.

Este artículo consta de las siguientes secciones: un apartado que describe algunos sistemas existentes de telerehabilitación, una sección en la que se explica la arquitectura del sistema propuesto y los módulos que la componen y finalmente las conclusiones.

2 Trabajos relacionados

Entre los distintos trabajos que se pueden encontrar en la literatura, relacionados tanto con el seguimiento visual como con el no visual, destacamos los siguientes:

Martin-Moreno propone en [7] una solución de rehabilitación basada en el mando Wii Remote de Nintendo. Este dispositivo es inalámbrico y usa un acelerómetro para registrar los movimientos del usuario en tres dimensiones. El sistema consiste en dos

módulos, uno con el que interactúan los usuarios y otro para el fisioterapeuta. El módulo del usuario ofrece una interfaz intuitiva que permite seleccionar qué ejercicios realizar y además indica si está realizando o no correctamente el ejercicio. La interfaz del fisioterapeuta permite el acceso remoto a los datos de la persona, consultar los resultados y modificar el tratamiento si lo considera oportuno.

Otro sistema de telerehabilitación lo propone Holden en [4]. En este caso se usa un dispositivo de seguimiento electromagnético. El sistema muestra a la persona el ejercicio a realizar, y esta debe imitarlo usando el dispositivo de seguimiento. Un algoritmo evalúa el ejercicio presentado y el del usuario, en función de distintos parámetros como velocidad, tiempo o precisión. Una opción del sistema es iniciar una videoconferencia con el fisioterapeuta y transmitirle, en tiempo real, los datos de los ejercicios. Los ejercicios se cargan mediante scripts que semi-automatizan las terapias y todos los datos de las sesiones se almacenan en una base de datos que el fisioterapeuta puede consultar. Las pruebas realizadas con este sistema mostraron que se habían producido mejoras significativas en el estado de los usuarios que usaron el sistema.

Biotrak [6] es un sistema de rehabilitación para realizar tareas orientadas a juegos. Su objetivo es lograr un entorno virtual en el que personas con deterioro cognitivo puedan realizar ejercicios simples. El sistema de seguimiento está compuesto de dos cámaras de infrarrojos y marcadores reflectantes que proporcionan información tridimensional de los movimientos de los usuarios. Biotrak se compone de tres módulos: gestión, base de datos y ejercicios. El sistema proporciona una experiencia inclusiva que evade al usuario de la terapia motivándole para seguir con la rehabilitación.

Frente a sistemas como los anteriores, que utilizan sensores de inercia, acelerómetros o sistemas de tracking mediante marcado, Kinect ofrece una clara ventaja; la persona no debe ponerse ningún tipo de dispositivo ni prenda especial para interactuar con el sistema. Se consigue una interacción no invasiva y totalmente natural para el usuario, ideal para realizar ejercicios de rehabilitación. Así mismo agiliza la puesta en marcha del sistema al reducirse el número de elementos necesarios.

KineRehab es un sistema de terapia ocupacional basado en Kinect [1]. Permite realizar tres ejercicios diferentes: levantar los brazos al frente, levantar los brazos a los lados y levantar los brazos hacia arriba. El sistema detecta la posición de las articulaciones del usuario y establece si se ha alcanzado el objetivo. De esta forma se puede evaluar la precisión de los ejercicios realizados. Además de almacenar los datos de los ejercicios realizados, el sistema motiva al usuario mediante sonidos y vídeo. El sistema se evaluó dividiendo las pruebas en dos fases que se repetían dos veces alternas. Una primera fase en la que el fisioterapeuta indicaba al usuario cómo realizar el ejercicio y este lo repetía. Y una segunda fase donde la persona usaba KineRehab, sin intervención del fisioterapeuta. Los ejercicios realizados correctamente en la fase 2 con KineRehab se incrementaron significativamente respecto a los de la fase 1. Por otra parte los usuarios disfrutaron con la terapia y mostraron interés en seguir usando el sistema.

En la tabla 1 se resumen los sistemas presentados anteriormente y se puede apreciar que sus características son muy variadas. Nuestra propuesta va encaminada hacia el uso de Kinect pero incluye nuevos retos frente a los considerados por KineRehab. Cabe destacar que los sistemas analizados no disponen de una gran variedad de ejercicios. Nuestro objetivo es también trabajar con ejercicios de rehabilitación variados, como

ejercicios para los miembros superiores, ejercicios de cuerpo entero o ejercicios que se realizan sentado.

	Sistema de seguimiento	Ejercicios	Interfaz para el fisio	Comunicación de video con el fisio	Evaluación automática	Pruebas con pacientes
WiiTherapy [7]	Acelerómetro	Miembros superiores	Sí	No	Comprueba repeticiones y si se alcanzan ciertos puntos	Sí
Sistema de telerehabilitación [4]	Electromagnético	Miembros superiores	Sí	Sí	Evaluación de trayectorias	Sí
Biotrack [6]	Multitracking (Cámaras de infrarojos con marcadores)	Cuerpo completo (6 juegos)	Sí	No	Comprueba que los ejercicios se hayan realizado correctamente	Sí
KineRehab [1]	Kinect (Cámara de profundidad basada en infrarrojos)	Miembros superiores (3 ejercicios)	Sí	No	Comprueba repeticiones y si se alcanzan ciertos puntos	Sí

Tabla 1. Comparativa de sistemas de rehabilitación

Además, nuestra propuesta incluye dos aplicaciones novedosas. Una aplicación para el fisioterapeuta que le permite definir nuevos ejercicios de manera fácil en base a unos movimientos básicos. Estos nuevos ejercicios además pueden ser evaluados automáticamente en la aplicación cliente. Por otro lado, en las sesiones reales las terapias no suelen ser estáticas, se adaptan en función de los resultados del usuario. Con el objetivo de facilitar el análisis de los ejercicios y optimizar la terapia, se propone otra aplicación, un planificador alojado en el servidor, que realiza una gestión automática de las terapias. Esta aplicación analiza los resultados de los usuarios y basándose en protocolos y datos médicos previos sugiere al fisioterapeuta los cambios más adecuados para la terapia del usuario.

3 Arquitectura del sistema y módulos

Un sistema de telerehabilitación abarca funciones muy variadas, por ello la arquitectura que se ha diseñado está dividida en módulos y con estructura cliente-servidor, como se muestra en la figura 1. Se hace distinción entre el cliente usuario y el cliente fisio que acceden al mismo servidor pero ofrecen distinta funcionalidad.

En el programa cliente del usuario los datos obtenidos de Kinect se estructuran para realizar la evaluación de los ejercicios realizados. El módulo de procesamiento de datos trata los datos recibidos y crea un descriptor de la pose del usuario. El modulo de evaluación tras generar los descriptores evalúa los ejercicios realizados y establece si se han ejecutado de manera correcta comparando los resultados obtenidos con los datos esperados para ese tipo de ejercicio. A la hora de mostrar a la persona los ejercicios que debe realizar es necesario que la visualización sea atractiva y que anime a la persona a participar en la terapia. El modulo de visualización genera un avatar en 3D que realiza el ejercicio correspondiente mostrando a la persona cómo debe hacerlo. Esta visualización puede incluir comentarios y consejos para que el usuario comprenda los objetivos del ejercicio. Por otro lado el usuario puede visualizar en otro avatar durante la ejecución de un ejercicio los movimientos que él mismo realiza e indicaciones para corregir errores. Como se ha dicho anteriormente Kinect incluye una cámara de video adecuada

para la realización de videoconferencias y micrófonos. El módulo de comunicación exterior gestiona las comunicaciones con el fisioterapeuta así como la transmisión de información a la base de datos del servidor. En el programa cliente del fisioterapeuta destaca la herramienta de gestión de terapias que es la que permite introducir y configurar ejercicios en el sistema y gestionar las terapias de los usuarios.

Por último en el servidor se encuentra el módulo de planificación automática, una de las características más innovadoras de este sistema, se encarga de modificar automáticamente las terapias que debe realizar la persona en función de los resultados del mismo y de acuerdo a una base de conocimiento establecida con ayuda de fisioterapeutas, para así obtener unos resultados óptimos.

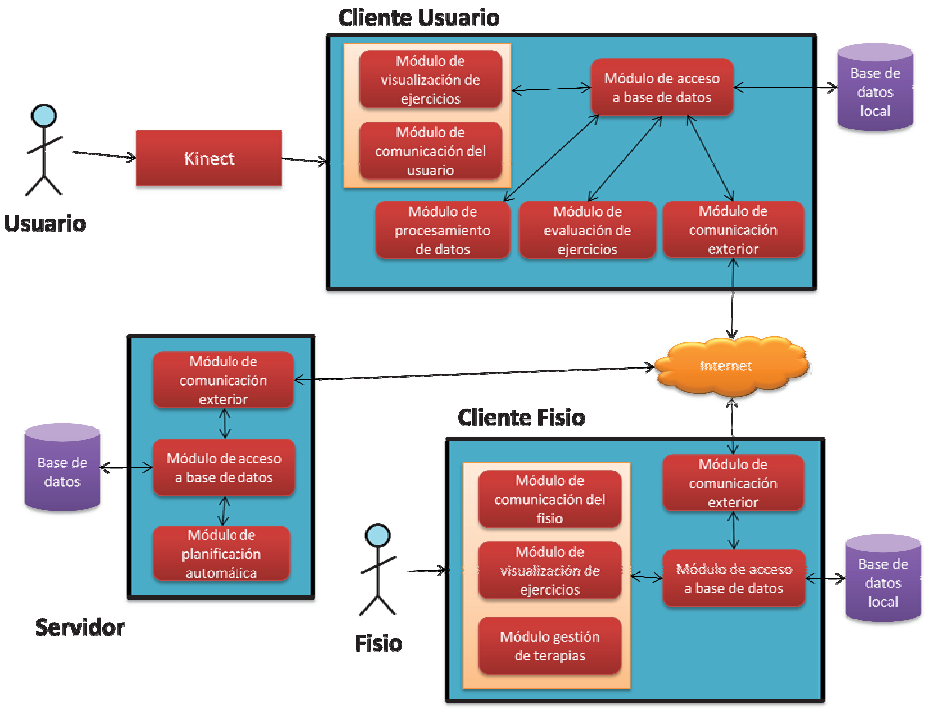


Fig. 1. Arquitectura del sistema

4 Conclusiones

En la actualidad es un hecho reconocido que las funcionalidades de los sistemas de telerehabilitación son limitadas. En muchos casos los ejercicios disponibles en ellos son reducidos y la inclusión de nuevos ejercicios no es lo bastante sencilla como para que sea el fisioterapeuta por sí mismo quien lo realice.

Un reto que se aborda en este proyecto consiste en permitir incorporar nuevos ejercicios de una manera simple, lo que implica, además, el desarrollo de una aplicación que es capaz de evaluarlos correctamente. Por otra parte, en las terapias reales, el fisio-

terapeuta no establece una terapia fija que sigue de manera estricta. Generalmente la terapia se adapta a los resultados de la persona, prolongando, reduciendo o añadiendo nuevos ejercicios cuando resulte necesario. La modificación automática de la terapia o la capacidad del sistema para sugerir cambios es otro aspecto que no se ha abordado en los sistemas de telerehabilitación actuales y que se considera en este proyecto.

En definitiva nuestra propuesta de sistema de telerehabilitación cuenta con un sistema de control totalmente natural para el usuario, es flexible en la definición de terapias e incluye aplicaciones de evaluación y gestión con el fin de simplificar y automatizar al máximo su funcionamiento. Actualmente el prototipo gestiona la recepción de datos de Kinect y los procesa para identificar y evaluar los movimientos de los ejercicios. El resto de módulos se encuentran en fase de desarrollo para lo que contamos con la colaboración de un grupo de fisioterapeutas que nos están proporcionando ejercicios y datos de terapias así como asesoramiento, desde la perspectiva clínica, de la gestión, evaluación y tratamiento del usuario [2]. El objetivo último es disponer de un sistema funcional y llevar a cabo pruebas reales para evaluar las características y la utilidad práctica del sistema.

Referencias

1. Chang, Y., Chen, S., & Huang, J. (2011). A kinect-based system for physical rehabilitation: A pilot study for young adults with motor disabilities. Research in Developmental Disabilities, 32(6), 2566-2570. doi:10.1016/j.ridd.2011.07.002
2. Buckup, K. (1996). Pruebas clínicas para patología ósea, articular y muscular: Exploraciones, signos, síntomas Masson.
3. Hanif, M., Niaz, H., & Khan, M. A. (2011). Investigating the possible role and usefulness of video capture virtual reality in motor impairment rehabilitation. Paper presented at the Next Generation Information Technology (ICNIT), 2011 the 2nd International Conference on, pp. 23-30.
4. Holden, M. K., Dyar, T. A., & Dayan-Cimadoro, L. (2006). Design and testing of a telerehabilitation system for motor re-training using a virtual environment. Paper presented at the Virtual Rehabilitation, 2006 International Workshop on, pp. 134-139.
5. Kinect for Windows.(2012). Retrieved from http://www.microsoft.com/en-us/kinectforwindows/discover/features.aspx (18/04/2012)
6. Llorens, R., Gil-Gomez, J., Mesa-Gresa, P., Alcaniz, M., Colomer, C., & Noe, E. (2011). BioTrak: A comprehensive overview. Paper presented at the Virtual Rehabilitation (ICVR), 2011 International Conference on, pp. 1-6.
7. Martin-Moreno, J., Ruiz-Fernandez, D., Soriano-Paya, A., & Jesus Berenguer-Miralles, V. (2008). Monitoring 3D movements for the rehabilitation of joints in physiotherapy. Paper presented at the Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE, pp. 4836-4839.
8. Rizzo, A. S., & Kim, G. J. (2005). A SWOT analysis of the field of virtual reality rehabilitation and therapy. Presence: Teleoperators & Virtual Environments, 14(2), 119-146.
9. Weiss, P. L., Rand, D., Katz, N., & Kizony, R. (2004). Video capture virtual reality as a flexible and effective rehabilitation tool. Journal of NeuroEngineering and Rehabilitation, 1(1), 12.
10. Zhou, H., & Hu, H. (2008). Human motion tracking for rehabilitation--A survey. Biomedical Signal Processing and Control, 3(1), 1-18.

Integración de observaciones medioambientales: Solución inicial y retos futuros *

Manuel A. Regueiro, Sebastián Villarroya, Gabriel Sanmartín, and José R.R. Viqueira

Grupo de Gráficos por Computador e Ingeniería de Datos (COGRADE),
Instituto de Investigaciones Tecnológicas,
Universidade de Santiago de Compostela
Constantino Candeira S/N, Santiago de Compostela
manuelantonio.regueiro@usc.es
sebastian.villarroya@usc.es
gabriel.sanmartin@usc.es
jrr.viqueira@usc.es

Resumen. En este trabajo se presenta una solución inicial para la integración de fuentes de datos de observación en aplicaciones de tipo medioambiental. La solución se basa en la utilización de una arquitectura típica mediador/wrapper combinada con modelos de datos e interfaces estándar definidos en la iniciativa Sensor Web Enablement (SWE) del Open Geospatial Consortium (OGC). Los retos futuros van en la dirección de simplificar la incorporación de nuevas fuentes de datos en el sistema, simplificando el desarrollo de los wrappers y proporcionando mecanismos más avanzados para la definición de relaciones semánticas entre elementos locales y globales.

Palabras clave: Integración de datos, Sensor Web, Gestión de datos Medioambientales, Arquitecturas Mediador/Wrapper, Interoperabilidad Espacial

1 Introducción

En la actualidad existe una gran cantidad de datos generados diariamente por multitud de sensores relacionados con aplicaciones de gestión medioambiental. Tanto los modelos y formatos utilizados para el almacenamiento de estos datos como las interfaces implementadas para su acceso son completamente heterogéneas y en muchos casos basadas en tecnologías propietarias. Por otro lado, diversas iniciativas promulgadas por las distintas administraciones públicas incluyen como objetivo la creación de Infraestructuras de Datos Espaciales (IDEs) que faciliten el acceso a los datos arriba mencionados. Muchas de estas iniciativas están motivadas por la aparición de la directiva INSPIRE de la Comisión

* Este trabajo ha sido financiado por la Xunta de Galicia (ref. 09MDS034522PR) y el Ministerio de Ciencia e Innovación, Gobierno de España (ref. TIN2010-21246-c02-02)

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 53-58, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

Europea. La interoperabilidad entre los distintos componentes y servicios de estas IDEs se basa en el uso de estándares de modelos, lenguajes e interfaces de servicios web. En el caso de los datos de sensores en entornos medioambientales, se utilizan fundamentalmente los estándares propuestos por el Open Geospatial Consortium (OGC) en su iniciativa Sensor Web Enablement (SWE). El acceso a fuentes de datos de sensores de tipo heterogéneo a través de lenguajes e interfaces estandarizados es un caso específico del problema de integración de fuentes de datos heterogéneas estudiado ampliamente en el área de las bases de datos.

En este trabajo se presenta una solución inicial dada dentro del proyecto MeteoSIX, financiado por la Xunta de Galicia, para la integración de datos de observación de distintas fuentes disponibles en la agencia meteorológica gallega (MeteoGalicia), a través de la interfaz Sensor Observation Service (SOS) del OGC. Estas fuentes incluyen datos generados por estaciones meteorológicas y oceanográficas, datos de radar y radiosondaje. La solución propuesta utiliza una arquitectura clásica basada en el uso de un mediador y diversos wrappers. El modelo global utilizado para la integración de datos en el mediador está basado en el estándar Observations and Measurments (O&M) usado en la interfaz SOS del OGC. La solución utilizada para la transformación de las consultas sobre el modelo global a consultas sobre los modelos de las fuentes de datos locales se basa en una aproximación Global as View (GAV) sobre el lenguaje de consulta de observación proporcionado por el SOS.

El resto de este artículo se organiza de la siguiente manera. En la Sec. 2 se proporciona una breve descripción de los estándares SOS y O&M del OGC, así como algunas referencias a otros trabajos relacionados en el ámbito de la integración de datos. La solución propuesta y actualmente implementada en el proyecto MeteoSIX se proporciona en la Sec. 3. El artículo termina con una breve descripción de los retos futuros en el ámbito de la integración de datos mediante la interfaz SOS.

2 Trabajo Relacionado

El estándar O&M [6] del OGC define un modelo de datos y una codificación XML para la representación e intercambio de observaciones. En general, cada observación vincula un valor observado (distintos tipos de valores son posibles) con un instante de tiempo. Además, se recogen algunos metadatos que contextualizan el valor observado. Los más importantes son el procedimiento mediante el cual se ha obtenido el valor (*Process*), la propiedad observada (*Observed Property*) y la entidad sobre la cual se realiza la observación (*Feature of Interest - FOI*). Típicamente, un *Process* será un sensor para medición en local o remoto montado en alguna plataforma, estática o móvil. La propiedad más importante de la *FOI* es su localización geográfica. El estándar de servicio SOS [7] proporciona una interfaz para el acceso a datos y metadatos de observaciones. Las observaciones de un SOS se organizan en vistas llamadas *Offerings*. El conjunto de *Offerings* definidas en un SOS se puede obtener mediante la llamada a la operación *GetCapabilities* de su interfaz. La operación *GetObservation* obtiene

el conjunto de observaciones de una determinada *Offering* para una o varias propiedades observadas. Opcionalmente, el cliente puede filtrar el resultado especificando un conjunto de *Process*, un filtro temporal y un filtro espacial. Otras operaciones SOS permiten obtener datos adicionales de un determinado *Process* o FOI e insertar y borrar observaciones. En [2] se describen varios proyectos en los que se utilizan estándares SWE.

Dos grandes aproximaciones surgen para la integración de fuentes de datos de forma transparente al usuario. En un *Data Warehouse* los datos de varias fuentes se cargan en un modelo de datos común, proporcionando una solución eficaz en la que los datos pueden volverse obsoletos rápidamente. Una solución de este tipo para datos de observación se describe en [8]. Soluciones alternativas se basan en la integración virtual de las fuentes. Los problemas clave a resolver en este tipo de integracin virtual incluyen los siguientes [3]. En primer lugar es necesario definir un esquema global virtual en el que puedan integrarse todas las fuentes. En segundo lugar hay que especificar la relación entre el esquema global y cada uno de los esquemas locales. En este punto es importante la resolución de conflictos tanto sintácticos como semánticos. Dos grandes aproximaciones existen para esta tarea [3]: i) Global as View (GAV) en la que el modelo global se define mediante vistas sobre los esquemas locales, y ii) Local as View (LAV) en las que cada modelo local se define como una vista sobre el esquema global. Por último, se necesitan estratégias de procesamiento eficiente y optimización de consultas sobre fuentes distribuidas con capacidades heterogéneas. Trabajos relacionados existen en el ámbito de la integración de datos científicos [4] y en el modelado conceptual de datos de observación [1].

3 Solución Actual

En el diseño actual del sistema se han considerado cuatro fuentes de datos de observación disponibles en MeteoGalicia. La primera fuente almacena observaciones generadas por la red de más de 80 estaciones meteorológicas automáticas. La segunda fuente da acceso a las observaciones generadas por una pequeña red de estaciones oceanográficas. Los modelos de datos de estas dos fuentes han sido diseñados en MeteoGalicia y están implementados en un gestor Microsoft SQL Server. La tercera fuente consta de secuencias temporales de archivos de radar. Cada archivo almacena varias imágenes de precipitación la misma extensión geográfica gallega. La ltima fuente almacena información de radiosondaje en un gestor PostgreSQL. El radiosondaje consiste en la suelta de globos equipados con un GPS y varios sensores que miden temperatura, humedad, temperatura del rocío, presión atmosférica, etc.

El primer paso para la integración de las fuentes anteriores es la definición de un modelo de datos global para el mediador. Este modelo está basado en los estándares O&M y SOS y su esquema relacional se muestra en la Fig. 1. Cada relación **Off$_\mathbf{i}$_Obs** almacena las observaciones de la *Offering* i definida en el sistema. De cada observación se almacena su instante de tiempo (time), *Process* (procId), *Observed Property* (propId), valor observado (result), y datos de su *FOI*

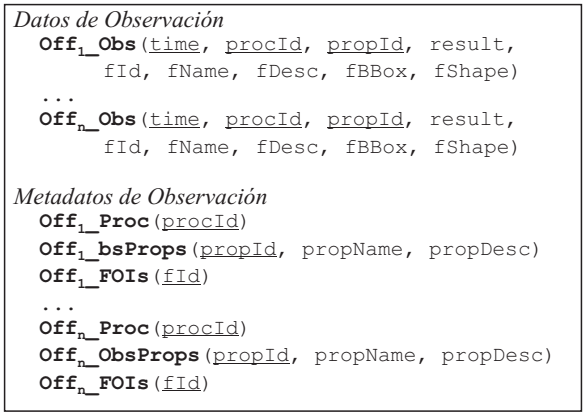


Fig. 1. Esquema Global.

que incluyen un identificador (fId, posiblemente nulo) y una geometría (fShape). Estos datos se utilizan para poder responder a las peticiones *getObservation*. Además de las observaciones, para cada *Offering* se almacenan los metadatos que la definen (procesos, propiedades observadas y FOIs). Estos metadatos se utilizan para poder responder a las peticiones *getCapabilities*.

Para cada fuente de datos se desarrolla un wrapper que implementa la interfaz SOS. Por lo tanto, cada una de las fuentes de datos tendrá el mismo modelo de datos que el usado como modelo global (ver Fig. 1). De acuerdo con lo anterior, el wrapper de cada fuente de datos S_i proporcionará para cada una de sus *Offerings* Off_j las relaciones $\mathbf{S_i\_Off_j\_Obs}$, $\mathbf{S_i\_Off_j\_Proc}$, $\mathbf{S_i\_Off_j\_ObsProps}$, $\mathbf{S_i\_Off_j\_FOIs}$, que dan acceso, respectivamente, a las observaciones, procesos, propiedades observadas y FOIs de la *Offering*. Además, para llevar a cabo el proceso de integración es necesario especificar la correspondencia entre cada proceso, propiedad observada y FOI local y su respectivo elemento global. Para conseguir esto, cada fuente de datos S_i debe de proporcionar las funciones *S_i_globalProc(procId)*, *S_i_globalProp(propId)* y *S_i_globalFOI(fId)*, que devuelven, respectivamente, el identificador global de proceso, propiedad observada y FOI para el correspondiente identificador local. Estas funciones proporcionan información semántica necesaria para la integración (ver "glue knowledge" en [4]). En la implementación actual de estas funciones, se asume que cada proceso y FOI de cada fuente de datos se incorpora al sistema global como un elemento distinto [1]. Por el contrario, varias propiedades observadas de varias fuentes de datos pueden incorporarse al sistema como una única propiedad observada global.

Para definir cada *Offering* Off_i del modelo global se especifican sus relaciones $\mathbf{Off_i\_Proc}$, $\mathbf{Off_i\_Prop}$ y $\mathbf{Off_i\_FOI}$. Además, se define un filtro temporal y un

[1] En concreto, el identificador local del elemento se concatena al identificador de la fuente de datos para obtener su identificador global

```
(1)Off_i_Obs = {time, procId, propId, result, fId, ..., fShape |
(2)         Off_i_Proc(procId) ∧  Off_i_Prop(propId, ...) ∧
(3)         temporalFilter_i(time) ∧
(4)         (spatialFilter_i(fShape) ∨ Off_i_FOI(fId)) ∧
(5)           (
(6)             (
(7)              S_1_Off_1_Obs(time, S_1_procId, S_1_propId, result,
(8)                            S_1_fId, ..., fShape) ∧
(9)              S_1_globalProc(S_1_procId) = procId ∧
(10)             S_1_globalProp(S_1_propId) = propId ∧
(11)             S_1_globalFOI(S_1_fId) = fId
(12)            )
(13)            ∨  . . .  ∨
(14)            (
(15)             S_n_Off_m_Obs(time, S_n_procId, S_1_propId, result,
(16)                           S_n_fId, ..., fShape) ∧
(17)             S_n_globalProc(S_n_procId) = procId ∧
(18)             S_n_globalProp(S_n_propId) = propId ∧
(19)             S_n_globalFOI(S_n_fId) = fId
(20)            )
(21)          )
(22)       }
```

Fig. 2. Definición de una *Offering* en el esquema global.

filtro espacial para la *Offering*. En base a estas definiciones las observaciones de Off_i se obtienen de las observaciones de las *Offerings* locales $\mathbf{S_i\_Off_j\_Obs}$ tal y como se especifica en la expresión de cálculo relacional de dominios de la Fig. 2. De forma muy breve, en la línea (2) se comprueba que la observación es de algún proceso y propiedad observada especificados para Off_i. Las líneas (3-4) chequean los filtros temporal y espacial. Por último, las líneas (7-19) verifican la existencia de alguna observación en alguna fuente de datos cuyos parámetros coincidan con los definidos globalmente para Off_i, haciendo uso de las funciones que especifican el enlace semántico entre elementos locales y globales.

No es difícil verificar que la consulta de la Fig. 2 se puede descomponer fácilmente en un conjunto de peticiones *GetObservation* a las distintas fuentes. Los filtros espacial y temporal pueden ser delegados en el wrapper de cada fuente, dependiendo de las capacidades de filtrado que implemente declaradas en su respuesta *GetCapabilities*.

4 Conclusiones y Retos Futuros

La solución actual asume la especificación por parte de las fuentes de datos de ciertas correspondencias semánticas con elementos del modelo global. Además, la implementación de los wrappers es excesivamente compleja debido a la transfor-

mación de modelos que deben de realizar. Líneas futuras a explorar para mejorar los problemas anteriores incluyen las siguientes:

- Reducir la funcionalidad de los wrappers limitándolos a transformar las consultas y datos a un lenguaje común. Una posible solución sería el uso de la interfaz Web Feature Service (WFS) del OGC que proporciona funcionalidad limitada de consulta sobre cualquier modelo de datos con información geográfica.
- Explorar la posibilidad de incorporar una solución LAV para la integración.
- Explorar el uso de ontologías para guiar el proceso de integración a nivel semántico [5]. Se trataría de simplificar la tarea del usuario a la hora de especificar las relaciones semánticas entre los elementos locales y globales. En la actualidad existen ya algunas ontologías conformes con el modelo O&M que se podrían usar [9].

References

1. Shawn Bowers, Joshua S. Madin, and Mark P. Schildhauer. A conceptual modeling framework for expressing observational data semantics. In *Proceedings of the 27th International Conference on Conceptual Modeling*, ER '08, pages 41–54, Berlin, Heidelberg, 2008. Springer-Verlag.
2. Helen Conover, Gregoire Berthiau, Mike Botts, H. Michael Goodman, Xiang Li, Yue Lu, Manil Maskey, Kathryn Regner, and Bradley Zavodsky. Using sensor web protocols for environmental data acquisition and management. *Ecological Informatics*, pages 32–41, 2010.
3. Alon Y. Levy. *Logic-based techniques in data integration*, pages 575–595. Kluwer Academic Publishers, Norwell, MA, USA, 2000.
4. Bertram Ludäscher, Amarnath Gupta, and Maryann E. Martone. A model-based mediator system for scientific data management. In *Bioinformatics: Managing Scientific Data*, pages 335–370. Morgan Kaufmann, 2003.
5. E. Mena, A. Illarramendi, V. Kashyap, and A. Sheth. OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. *International journal on Distributed And Parallel Databases (DAPD)*, 8(2):223–272, April 2000.
6. OGC. Observations and measurements part 1 - observation schema. Open Geospatial Consortium (OGC). Retrieved January 2011 from: http://www.opengeospatial.org, 2007.
7. OGC. Sensor observation service. Open Geospatial Consortium (OGC). Retrieved January 2011 from: http://www.opengeospatial.org, 2007.
8. José Ramon Rios Viqueira, José Varela, Joaquín A. Triñanes, and José Manuel Cotos. A sensor observation service based on ogc specifications for a meteorological sdi in galicia. In *ER Workshops*, pages 43–52, 2010.
9. W3C Semantic Sensor Network Incubator Group. Semantic sensor network xg final report. World Wide Web Consortium (W3C). Retrieved April 2012 from: http://www.w3.org/2005/Incubator/ssn/XGR-ssn/, 2011.

Análisis espacio-temporal en sistemas de bases de datos lógico-funcionales *

Sebastián Villarroya, Gabriel Álvarez, Roi Méndez, and José R.R. Viqueira

Grupo de Gráficos por Computador e Ingeniería de Datos (COGRADE),
Instituto de Investigaciones Tecnológicas,
Universidade de Santiago de Compostela
Constantino Candeira S/N, Santiago de Compostela
sebastian.villarroya@usc.es
gabriel.alvarez@usc.es
roi.mendez@usc.es
jrr.viqueira@usc.es

Resumen En este artículo se describe de forma muy breve un nuevo lenguaje para análisis de datos espacio-temporales. El lenguaje se basa en un modelo de datos funcional cuya estructura de datos (llamada *MappingSet*) almacena conjuntos de funciones con dominios idénticos. La sintaxis XML del lenguaje combina el cálculo relacional en la especificación de los dominios de los *MappingSets* y el paradigma funcional en la especificación de las funciones. La simplicidad del modelo facilita la implementación actual de un servicio web de análisis espacial.

Palabras clave: Bases de datos espacio-temporales, Modelado de datos, Lenguajes de consulta, Análisis espacio-temporal, Sistemas de Información Geográfica

1. Introducción

La gestión de datos espaciales y espacio-temporales desde dentro de los Sistemas Gestores de Bases de Datos (SGBDs) es un problema estudiado desde hace muchos años en el área de bases de datos espaciales. Multitud de soluciones se han propuesto en las que los datos de entidades espaciales, es decir, entidades que incorporan propiedades de tipo espacial, se gestionan con modelos y lenguajes de bases de datos. Estas soluciones han dado como fruto la aparición de estándares de uso extendido como son el SQL/MM de ISO y la Simple Feature Specification for SQL (SFS) del Open Geospatial Consortium (OGC). Una cobertura espacial es un conjunto de funciones que comparten el mismo dominio espacial. Así, una cobertura meteorológica asociará valores de temperatura, humedad, etc. a cada punto de la zona del espacio que define su dominio. La

* Este trabajo ha sido financiado por el Ministerio de Ciencia e Innovación, Gobierno de España (ref. TIN2010-21246-c02-02)

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 59-64, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

naturaleza funcional de las coberturas hace que su incorporación en los modelos y lenguajes lógicos que predominan en las bases de datos no resulte en soluciones elegantes. A pesar de esto, algunas soluciones existen que permiten la gestión de coberturas en bases de datos mediante constructores para la manipulación de arrays en modelos lógicos. Incluso en estos casos, el usuario tendrá que ser capaz de usar dos conjuntos distintos de estructuras y operadores, para entidades y coberturas, respectivamente.

En este trabajo se propone un modelo de datos y un lenguaje que permite el análisis de datos de entidades y coberturas de una forma totalmente homogénea. Tanto las entidades como las coberturas se modelan en el sistema como conjuntos de funciones de datos llamadas *MappingSets*. El análisis espacial y espacio-temporal se consigue con la definición de *MappingSets* derivados mediante un lenguaje declarativo que combina formalismos lógicos y funcionales. Este lenguaje incorpora tipos de datos espaciales tanto para la representación de figuras espaciales (puntos, líneas y superficies) como para la definición de muestreos del espacio a distintas resoluciones. Tanto la precisión utilizada en la representación de las figuras como la precisión y resolución de los muestreos son controladas por el usuario mediante parámetros de los tipos de dato.

El resto del artículo se organiza como sigue. En la Sec. 2 se proporciona un descripción de algunos trabajos relacionados con este. El modelo y lenguaje de análisis espacio-temporal se ilustra en la Sec. 3. El artículo termina con unas breves conclusiones y retos de trabajo futuro.

2. Trabajo relacionado

Los Sistemas de Información Geográfica (SIG) gestionan dos grandes tipos de información, en concreto, datos sobre *Entidades* y *Coberturas Espaciales*. Una *Entidad Espacial* nos es más que una entidad de un determinado dominio de aplicación que tiene alguna propiedad de tipo geométrico (puntos, líneas y superficies). Ejemplos de *Entidades Espaciales* son municipios, ríos, carreteras, etc. El modelado y gestión de este tipo de información en paradigmas tradicionales de bases de datos ha sido un tema de investigación ampliamente tratado durante muchos años. En general, modelos como el relacional son extendidos con tipos de dato que permiten la representación de geometrías y predicados y funciones que incorporadas en lenguajes como el SQL permiten su gestión. Como resultado de estos trabajos aparece la parte espacial del estándar SQL Multimedia [4], que define tipos de dato SQL:2003 para datos espaciales representados mediante geometrías vectoriales. En cuanto a la evolución temporal de estos datos, en [6] se combinan tipos de dato espaciales y temporales con nuevas operaciones primitivas del modelo relacional. La principal característica de esta aproximación es que consigue la gestión de datos espacio-temporales con un conjunto muy reducido de operadores. La evolución continua en el tiempo de objetos espaciales se gestiona en [3] mediante la definición de tipos de dato espacio-temporales para "Moving Objects".

Una *Cobertura Espacial* es un conjunto de funciones que tiene un dominio espacial común. Por ejemplo, una *Cobertura* meteorológica definida sobre una determinada área geográfica tendrá funciones que asignan a cada punto de dicha área valores de temperatura, humedad relativa, presión atmosférica, etc. La implementación de este tipo de información en un SIG se basa normalmente en la utilización de algún tipo de mosaico regular para discretizar el espacio (generalmente dividiéndolo en celdas cuadradas), y almacenando los valores de las funciones para cada celda (representaciones Raster del espacio). Al contrario de las *Entidades Espaciales*, las coberturas en un SIG generalmente no se almacenan en bases de datos, sino en archivos con formatos de imagen o con formatos específicos para datos de este tipo como son el NetCDF. El hecho de que el almacenamiento explícito de cada uno de los elementos del dominio de una cobertura sea totalmente ineficiente hace que su gestión directa en bases de datos no sea siquiera una opción a considerar. A pesar de esto, algunas pocas soluciones gestionan coberturas en bases de datos mediante nuevos tipos de datos para el almacenamiento de datos raster [5] o arrays [7]. En [1] se describe un sistema diseñado específicamente para la gestión de arrays en entornos científicos.

En cualquiera de los casos anteriores, la gestión uniforme de Entidades y Coberturas Espaciales es o imposible o requiere del uso de dos conjuntos de tipos de datos y funciones por parte del usuario. La existencia de muchas aproximaciones en el área de bases de datos funcionales [2] para la gestión de entidades y la naturaleza funcional de las coberturas hacen que en el presente trabajo se aborde el problema de la gestión uniforme de todos estos datos dentro de un paradigma funcional.

3. Descripción del lenguaje

El lenguaje dispone de los siguientes tipos de dato. Los tipos **Boolean**, **CString** y **Numeric**(D,P) tienen su semántica convencional. El tipo de dato **NumericSample**(D,P,R) representa los elementos de un proceso de muestreo sobre el espacio 1D definido por el tipo **Numeric**(D,P) usando una resolución de muestreo de $R * 10^{-P}$ y una fase de muestreo H que depende de la implementación. La Fig. 1(a) ilustra los tipos **Numeric**(2,1) y **NumericSample**(2,1,5), usando una fase $H = 0{,}2$. Los tipos de dato **TimeInstant**(P) y **Time**(P) representan instantes de tiempo de la forma convencional. El sistema proporciona tipos para muestreos temporales **TimeInstantSample**(P,R) y **Time**(P,R). El tipo de dato **Date**, que representa fechas, se proporciona como un alias del tipo **TimeInstantSample**(0,86400). Los tipos de dato **Point2D**(D,P) y **Point3D**(D,P) representan puntos en 2 y 3 dimensiones, respectivamente, donde las coordenadas cartesianas de cada punto son de tipo **Numeric**(D,P). Los tipos de dato **Point2DSample**(D,P,R) y **Point3DSample**(D,P,R) representan los elementos de procesos de muestreo en espacios de dimensión 2 y 3 respectivamente. Las direcciones D1 y D2 en las que se realiza el muestreo y las fases para cada una de las direcciones dependen de la implementación. La resolución de muestreo $R * 10^{-P}$ será la misma en ambas direcciones. La Fig. 1(b) ilustra dos procesos

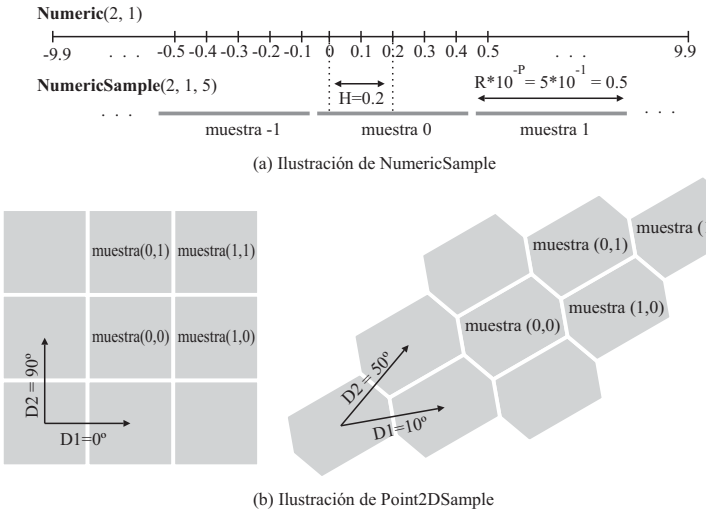


Figura 1. Ilustración de los tipos de dato NumericSample y Point2DSample.

de muestreo en espacios de dimensión 2 con direcciones de muestreo D1 y D2 distintas. El constructor de tipos **Interval**[T] genera un tipo de dato intervalo sobre elementos del tipo T (tipos numéricos y temporales, incluyendo tipos sample). Para un tipo S espacial (**Point2D**, **Point2DSample**, etc.), el sistema proporciona constructores para generar tipos geométricos sobre el tipo espacial. Por ejemplo, **LineString**(S) representa polilíneas sobre los puntos de S. **Polygon**(S) representa polígonos sobre los puntos de S. En general se proporcionan los tipos geométricos definidos en el estándar SQL/MM [4].

Un *MappingSet* es un conjunto de funciones (*Mappings*) que tienen como dominio común el conjunto de tuplas de una relación, en el sentido del modelo relacional. Así por ejemplo, el *MappingSet* "Municipality" tiene como dominio una relación con un único atributo mId de tipo **Numeric** que almacena identificadores de municipios. La función "name" de "Municipality" asocia a cada identificador de municipio su nombre de tipo **CString**. Este *MappingSet* puede tener otras funciones para almacenar la población y la geometría (**Polygon**[**Point2D**(D,P)]) de cada municipio. De forma similar, un *MappingSet* "Station" puede tener en su dominio identificadores de estaciones meteorológicas y funciones para almacenar en cada estación los últimos valores de varios parámetros meteorológicos (temperatura, humedad, etc.). La localización de cada estación puede ser una función más. Un último ejemplo sería una cobertura meteorológica "Meteo"podría tener en su dominio valores de tipo **Point2DSample** y funciones para almacenar valores de temperature, humedad, etc.

El lenguaje permite definir *MappingSets* de varios tipos. Los *MappingSets* persistentes definidos serán almacenados en la memoria persistente del sistema, mientras que los temporales serán utilizados durante una determinada transacción de análisis y luego serán eliminados. Los *MappingSets* persistentes pueden

```
<PersistentMappingSet name="app:newMunicipality" result="true">
 <RelationalDomain scheme="mId">
  <Relation  fromDomains="app:Municipality(mId AS m)">
   <Where>app:Municipality.population(m) &gt;= 10000</Where>
   <Return>m</Return>
  </Relation>
 </RelationalDomain>
 <DerivedMapping name="temp">
   <Aggregate fromDomains="app:Meteo(loc)">
     <Where>within(loc,app:Municipality.geometry(mId))</Where>
     <Return>AVG(app:Meteo.temperature(loc))</Return>
   </Aggregate>
 </DerivedMapping>
</PersistentMappingSet>

<PersistentMappingSet name="app:MeteoIDW">
  <ExtensionalDomain scheme="loc">
    <Point2DSampling>
      <Size dim1="45" dim2="45"/>
      <Origin><gml:pos>-71.106216 42.366661</gml:pos></Origin>
      <OffsetVector orientation="0" resolution="10"/>
      <OffsetVector orientation="90" resolution="10"/>
    </Point2DSampling>
  </ExtensionalDomain>
  <IntensionalMapping name="temperature">
   <Aggregate fromDomains="app:Stations(sId)">
    <Where>distance(loc,app:Stations.position(sId)) &lt; 2000</Where>
    <Return>
         SUM(app:Stations.temperature(sId)/
                distance(loc,app:Stations.position(sId))) /
         SUM(1/distance(loc,app:Stations.position(sId)))
    </Return>
   </Aggregate>
  </IntensionalMapping>
</PersistentMappingSet>
```

Figura 2. Ejemplo de uso de STAML para definir *MappingSets*.

ser primitivos, cuyos dominios y rangos son almacenados, o derivados en los que se almacena la definición de dominios y rangos, y el *MappingSet* se calcula cada vez que es necesario (concepto equivalente al de vista en SQL). El dominio de un *MappingSet* puede especificarse de forma extensional, especificando cada una de sus tuplas, o los parámetros que definen el muestreo en el caso de ser un dominio de tipos Sample. Además, el dominio de un *MappingSet* puede especificarse también mediante una sintaxis basada en el cálculo relacional de dominios. Las funciones de un *MappingSet* pueden definirse de forma extensional, especificando sus valores en el mismo orden en que han sido especificados los valores de dominio, o intensional, dando una expresión funcional que calcula los valores para cada elemento del dominio. Una función de un *MappingSet* especificada de forma intensional podrá ser definida como primitiva (su valor se calcula y almacena) o derivada (su valor se calcula cada vez que es necesario).

En la Fig. 2 se muestran dos ejemplos de definiciones de *MappingSet* especificando dos procedimientos de análisis espacial. El primer *MappingSet* obtiene la media de temperatura en cada municipio a partir de los *MappingSets* "Municipality" y "Meteo". El tag <**Relation**> se utiliza para expresar en forma de calculo relacional la consulta que devuelve el dominio del resultado. El tag

<**Aggregate**> permite evaluar funciones de agregado sobre un conjunto. El segundo *MappingSet* "MeteoIDW" obtiene una cobertura meteorológica aplicando el método de interpolación IDW (Inverse Distance Weight) sobre los datos de las estaciones meteorológicas. Es importante resaltar la forma en la que se especifica el muestreo que define el dominio de este segundo resultado.

4. Conclusiones y trabajo futuro

En este trabajo se ha descrito un lenguaje para la especificación declarativa de procedimientos de análisis espacial y espacio-temporal sobre datos de entidades y coberturas. El lenguaje utiliza un modelo de datos funcional. Se utilizan formalismos lógicos (cálculo relacional de dominios) para definir los dominios de los *MappingSets* y el paradigma funcional para definir sus funciones. En ente momento se está realizando la implementación de un servicio de análisis espacio-temporal de datos, cuya interfaz está basada en el lenguaje aquí descrito. El uso de funciones de tipos atómicos hace que esta implementación pueda utilizar estructuras de datos simples para almacenar los *MappingSets*. En concreto, actualmente cada *MappingSet* se almacena en un archivo con formato NetCDF. Las líneas de trabajo futuro incluyen la definición de estructuras de datos y métodos de acceso para entidades y coberturas, la inclusión en el servicio de funcionalidad de integración de datos y el procesamiento eficiente de las definiciones de *MappingSets* sobre fuentes de datos distribuidas disponibles a través de otros servicios web estándar.

Referencias

1. Paul G. Brown. Overview of scidb: large scale array storage, processing and analysis. In *Proceedings of the 2010 international conference on Management of data*, pages 963–968, New York, NY, USA, 2010. ACM.
2. Peter M.D. Gray, Larry Kerschberg, Peter J.H. King, and Alexandra Poulovassilis. *The Functional Approach to Data Management: Modeling, Analyzing and Integrating Heterogeneous Data*. Springer, Berlin, Heidelberg, New York, 2004.
3. R. H. Güting, M. H. Böhlen, M. Erwig, C. S. Jensen, N. A. Lorentzos, M. Schneider, and M. Vazirgiannis. A foundation for representing and querying moving objects. *ACM Transactions on Database Systems*, 25(1):1–42, 2000.
4. ISO. Sql multimedia and application packages (sql/mm) part 3: Spatial. iso/iec jtc 1/sc 32/wg 4: Vie-008. International Organization for Standardization (ISO), 2002.
5. Alejandro A. Vaisman and Esteban Zimányi. A multidimensional model representing continuous fields in spatial data warehouses. In *17th ACM SIGSPATIAL*, pages 168–177. ACM, November 4-6 2009.
6. J.R.R. Viqueira and N.A. Lorentzos. Sql extension for spatio-temporal data. *The VLDB Journal*, 16(2):179–200, 2007.
7. N. Widmann and P. Baumann. Towards comprehensive database support for geoscientific raster data. In *Proc. of the 5th ACM International Symposium on Advances in Geographic Information Systems*, pages 54–57. ACM Press, 13-14 November 1997.

Social Pathway Annotation: Extensions of the Systems Biology Metabolic Modelling Assistant

Ismael Navas-Delgado[*], Alejandro del Real-Chicharro[+], Miguel Ángel Medina[+], Francisca Sánchez-Jiménez[+], José F. Aldana-Montes[*]

[*]Computer Languages and Computing Science Department, University of Malaga, Spain.
[+]Molecular Biology and Biochemistry Department. University of Malaga, Spain.

Abstract. High-throughput experiments have produced large amounts of heterogeneous data in the life sciences. The integration of data in the life sciences is a key component in the analysis of biological processes. These data may contain errors, but the curation of the vast amount of data generated in the "omic" era cannot be done by individual researchers. To address this problem, community-driven tools could be used to assist with data analysis. In this paper, we focus on a tool with social networking capabilities built on top of the SBMM (Systems Biology Metabolic Modelling) Assistant to enable the collaborative improvement of metabolic pathway models (the application is freely available at http://sbmm.uma.es/SPA).

Keywords: Social Data Curation, Life Sciences, Data Integration

1 Introduction

Individual users can make single improvements of different metabolic pathways based on their experience. Collectively, a community of scientific users could provide highly curated metabolic pathways based on the community's experience. In this sense, social networking is a new issue that is attracting a lot of interest in many domains. In the case of biology, research networks have usually been closed ones, in which the most prominent researchers decided the roadmap of this research. Currently, new technologies are opening up new opportunities for knowledge exchange by means of social networks, creating new knowledge-driven digital communities.

The System Biology Metabolic Modelling Assistant (http://sbmm.uma.es) [1] is a tool developed to search, visualise, manipulate and annotate both identity data and kinetic data. Metabolic Modelling is attracting a lot of interest due to its potential to study biological processes from a systems biology point of view.

2 System Description

In this paper, we present Social Pathway Annotation (SPA, http://sbmm.uma.es/SPA), an extension of the SBMM Assistant, as community-based approach to the curation

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 65-66, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

process. This extension includes the following relevant tools for this community of users:

- A bibliographic tool that allows users to discover scientific networks through the links between papers related to a metabolic pathway, reaction or biological component.
- A curation tool that enables the editing of metabolic pathway elements (metabolites, reactions and their kinetics) by individual users.
- A social network tool that provides users a way to store metabolic pathway models and collaboratively curate them.

3 Discussion and Conclusions

Curation tools must evolve towards social curation, due to the impossibility (both in terms of people and resources) of managing the huge quantity of constantly increasing biological information provided by experimental methodologies. We have developed a novel tool that tests some basic theoretical aspects of the future of data curation, based on metabolic pathway curation. The advantages of this solution in comparison to a conventional metabolic pathway curation are as follows:

- Many eyes are able to look for model inconsistencies, whereas in conventional curation only a few eyes look for those inconsistencies.
- The controlled creation of models and versions allows the production of new ordered knowledge, which would not be the case if a lot of files were shared as an unordered amalgam of names.
- The use of a controlled vocabulary allows errors to be kept to a minimum in curation and also helps with the organisation, maintenance and querying of the constantly incoming metabolic pathway models.
- The extension over an easy-to-use, albeit powerful, and graphical user interface facilitates the curation task for inexperienced users.

4 Acknowledgements

The Project Grant TIN2011-25840 (Spanish Ministry of Education and Science) and P07-TIC-02978 and P11-TIC-7529 (Innovation, Science and Enterprise Ministry of the regional government of the Junta de Andalucía) have supported this work.

References

1. Reyes-Palomares A, Montanez R, Real-Chicharro A et al. Systems biology metabolic modeling assistant: an ontology-based tool for the integration of metabolic data in kinetic modeling, Bioinformatics 2009;25:834-835.

Una estructura Métrica Genérica para Búsquedas por Rango sobre una Plataforma Multi-GPU

Roberto Uribe-Paredes[1], Enrique Arias[2], Diego Cazorla[2], José L. Sánchez[2]

[1] Departamento de Ingeniería en Computación, Universidad de Magallanes, Punta Arenas, Chile.
[2] Departamento de Sistemas Informáticos, Universidad de Castilla La Mancha, Albacete, España.
e-mail: roberto.uribeparedes@gmail.com;
{enrique.arias,jose.sgarcia,diego.cazorla}@uclm.es

Resumen Actualmente, la búsqueda por similitud en espacios métricos es de interés para la comunidad científica debido a sus múltiples campos de aplicación, como reconocimiento de patrones y recuperación de la información, entre otros. El uso de índices métricos proporciona un aumento en la eficiencia durante los procesos de búsqueda mediante la reducción del número de evaluaciones de distancia. Sin embargo, para aplicaciones reales donde el volumen de datos a procesar es masivo, el tiempo de resolución de una consulta es altamente costoso. En este sentido, el procesamiento paralelo permite disminuir los tiempos de búsqueda. Para este propósito, modernas plataformas basadas en GPU/multi-GPU proporcionan impresionantes ratios coste/rendimiento. En este artículo se presenta un análisis experimental de un conjunto de estructuras métricas en sus versiones secuenciales para posteriormente ser evaluadas bajo una plataforma basada en GPU. Finalmente la mejor alternativa es implementada sobre múltiples GPUs. Como conclusión, una estructura métrica genérica presenta las mejores propiedades para este tipo de plataforma logrando rendimientos que superan en 32 veces los tiempos obtenidos con la mejor versión secuencial.

Keywords: Búsqueda por Similitud, espacios métricos, procesamiento paralelo, plataformas multi-GPU

1. Introducción

La búsqueda de objetos similares sobre un gran conjunto de datos se ha convertido en una línea de investigación de gran interés. Aplicaciones de estas técnicas pueden ser encontradas en reconocimiento de voz e imagen, en problemas de minería de datos, detección de plagios, búsqueda de objetos sobre bases de datos multimedia y muchas otras.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 67-80, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

En la actualidad, la necesidad de almacenar y procesar grandes volúmenes de datos hace necesario aumentar el rendimiento en términos de tiempo de procesamiento. En general, se utilizan para ello diversas plataformas paralelas. En este sentido, nuevas tecnologías basadas en tarjetas gráficas (*Graphic Processor Unit, GPU*) permiten un alto nivel de paralelismo a un muy bajo coste.

El principal aporte de este trabajo es presentar la implementación de una estructura métrica basada en pivotes sobre una plataforma de multiples GPUs. Para ello, se analiza mediante una serie de experimentos, algunos aspectos de interés de un conjunto de estructuras. Finalmente se selecciona una versión genérica representativa de este tipo de estructuras, que permite aprovechar algunas ventajas ofrecidas por este tipo de dispositivos. Se muestran los resultados obtenidos para versiones secuenciales y sobre una GPU. Posteriormente se selecciona la mejor alternativa y se implementa sobre una plataforma con multiples GPUs.

El artículo está estructurado de la siguiente manera. Primero, las siguientes subsecciones introducen el concepto de búsqueda por similitud, describe la arquitectura y el modelo de programación de una GPU y resume el trabajo relacionado en esta línea de investigación. En la Sección 2 se describen brevemente las estructuras utilizadas en el trabajo y el detalle de la búsqueda sobre una estructura genérica basada en pivotes. La Sección 3 presenta resultados preliminares importantes para aplicarlos en la Sección 4 y 5 que corresponden a las implementaciones sobre una y multiples GPUs. Finalmente, las conclusiones y trabajo futuro son presentados en la Sección 6.

1.1. Búsqueda por Similitud

La similitud se modeliza en muchos casos interesantes a través de un espacio métrico, y la búsqueda de objetos más similares a través de una búsqueda por rango o de vecinos más cercanos. Un espacio métrico es un conjunto $\mathbb{X}$ con una función de distancia $d : \mathbb{X}^2 \rightarrow \mathbb{R}$, tal que $\forall x, y, z \in \mathbb{X}$, se deben cumplir las propiedades de: positividad ($d(x, y) \geq 0$ *and* $d(x, y) = 0$ *ssi* $x = y$), simetría ($d(x, y) = d(y, x)$) y desigualdad triangular ($d(x, y) + d(y, z) \geq (d(x, z))$.

Sobre un espacio métrico $(\mathbb{X}, d)$ y un conjunto de datos finito $\mathbb{Y} \subseteq \mathbb{X}$, se pueden realizar una serie de consultas. La consulta básica es la *consulta por rango*. Sea una consulta $x \in \mathbb{X}$, y un rango $r \in \mathbb{R}$. La consulta de rango alrededor de x con rango r es el conjunto de puntos $y \in \mathbb{Y}$, tal que $d(x, y) \leq r$. Un segundo tipo de consulta, que puede construirse usando la consulta por rango, es *los k vecinos más cercanos*. Sea una consulta $x \in \mathbb{X}$ *y un entero* k. Los k vecinos más cercanos a x son un subconjunto $\mathbb{A}$ de objetos de $\mathbb{Y}$, donde $|\mathbb{A}| = k$ y no existe un objeto $y \in \mathbb{A}$ tal que $d(y, x)$ sea menor a la distancia de algún objeto de $\mathbb{A}$ a x.

El objetivo de los algoritmos de búsqueda es minimizar la cantidad de evaluaciones de distancia realizadas para resolver la consulta. Los métodos para buscar en espacios métricos se basan principalmente en dividir el espacio empleando la distancia a uno o más objetos seleccionados. El no trabajar con las características particulares de cada aplicación tiene la ventaja de ser más general, pues los

algoritmos funcionan con cualquier tipo de objeto [1]. Dependiendo de la estructura y del tipo de espacio, las funciones de distancia pueden ser continuas o discretas.

Algunas de las estructuras basan la búsqueda en *pivotes* y otras en *clustering*. Los algoritmos basados en clustering dividen el espacio en áreas o planos, donde cada área tiene un centro y se almacena alguna información sobre el área que permita descartarla completa sólo con comparar la consulta con su centro.

Existen dos criterios para delimitar las áreas en las estructuras basadas en clustering, *hiperplanos* y *radio cobertor* (*covering radius*). El primero divide el espacio en particiones de *Voronoi* y determina el hiperplano al cual pertenece la consulta según a qué centro corresponde. El criterio de radio cobertor divide el espacio en esferas que pueden intersectarse y una consulta puede pertenecer a más de una esfera.

Los algoritmos de clustering son los mejores para búsquedas sobre espacios de alta dimensión. Ejemplos de estos son *BST*, *GHT*, *M-Tree*, *GNAT*, *EGNAT* y muchos otros [1].

En los métodos basados en pivotes, se selecciona un conjunto de pivotes y se precalculan las distancias entre los pivotes y todos los elementos de la base de datos. Durante la búsqueda, se calcula la distancia de la consulta a los pivotes y se usa la desigualdad triangular para descartar objetos, evitando el cálculo directo entre la consulta y el elemento.

Algunas estructuras de este tipo son: *LAESA* [2], *FQT* y sus variates [3], *Spaghettis* y sus variantes [4], *FQA* [5], *SSS-Index* [6] y otros.

Las estructuras de tipo árbol utilizan esta técnica en forma indirecta. El árbol se va construyendo tomando el nodo raíz como pivote y dividiendo el espacio de acuerdo a la distancia de los objetos al pivote. La búsqueda realiza un backtrack sobre el árbol y utiliza la desigualdad triangular para minimizar los subárboles.

Otros algoritmos, de tipo arreglo, hacen una implementación directa de este concepto, y se diferencian básicamente en su estructura extra para reducir el coste de CPU para encontrar los puntos candidatos, pero no en la cantidad de evaluaciones de distancia. Ejemplos de éstos son: *LAESA*, *Spaghettis* y algunas variantes de estos. En el presente trabajo los algoritmos implementados son todos basados en pivotes y de tipo arreglo, los cuales son más adaptables a la GPU.

1.2. Unidades de Procesamiento Gráfico

Actualmente las unidades de procesamiento gráfico (GPUs) disponen de un alto número de cores con un alto ancho de banda con memoria. Este tipo de dispositivos permite aumentar la capacidad de procesamiento respecto de las CPUs [7]. Una tendencia denominada *Computación de Propósito General sobre GPU* o GPGPU ha orientado la utilización de GPUs sobre nuevos tipos de aplicaciones.

Posiblemente, todavía en la actualidad, el mayor problema radique en la programación de este tipo de dispositivos. Actualmente, las empresas fabricantes de GPUs han propuesto nuevos lenguajes o extensiones para los lenguajes de alto nivel más utilizados. Un ejemplo es la propuesta de NVIDIA con CUDA[8], la

cual consiste en una plataforma software que permite utilizar las GPUs para aplicaciones de propósito general, con la capacidad de manejar un gran número de hilos.

En general, las GPUs utilizan todos sus recursos en tener la mayor cantidad de elementos de procesamiento y gran ancho de banda con memoria, lo que posibilita una potencia de cálculo muy superior al de las CPUs. Así, uno de los objetivos de las GPUs es aumentar la capacidad de procesamiento explotando el paralelismo a nivel de datos para problemas paralelos o que se puedan paralelizar.

Modelo de Programación CUDA El modelo de Programación CUDA de NVIDIA considera a la GPU como un dispositivo capaz de ejecutar un alto número de hilos en paralelo. Este modelo hace una distinción entre el código ejecutado en la CPU (host) con su propia DRAM (*host memory*) y el ejecutado en GPU (device) sobre su DRAM (*device memory*). CUDA incluye herramientas de desarrollo software C/C++, librerías de funciones, y un mecanismo de abstracción que oculta los detalles hardware al desarrollador.

En CUDA los hilos son organizados en bloques del mismo tamaño y los cálculos son distribuidos en una malla o grid de bloques. Estos hilos ejecutan el código de la GPU, denominado *kernel*. Un *kernel* CUDA ejecuta un código secuencial en un gran número de hilos en paralelo.

Los hilos en un bloque (actualmente 512 hilos por bloque y 1024 para la nueva arquitectura *Fermi* de NVIDIA) pueden trabajar juntos eficientemente, intercambiando datos localmente en la *memoria compartida* y sincronizando la baja latencia en la ejecución mediante puntos de sincronización llamados *barreras*. Por el contrario, lo hilos ubicados en diferentes bloques pueden compartir datos, solamente accediendo a la memoria global de la GPU o device memory, que es una memoria de alta latencia.

1.3. Trabajo Relacionado

Actualmente, existe una gran variedad de plataformas paralelas sobre las cuales se pueden implementar estructuras métricas. En este contexto, muchos estudios se han enfocado inicialmente a plataformas de memoria distribuida usando bibliotecas de alto nivel como MPI o PVM, y memoria compartida usando directivas de OpenMP [9].

Algunos estudios se han enfocado a la paralelización de diferentes estructuras sobre plataformas de memoria distribuida y a la distribución de los datos y el balance de carga usando MPI o BSP, usualmente plataformas del tipo clusters.

Menos son los trabajos orientados hacia aplicaciones de memoria compartida. En general, algunos estudios analizan la distribución de consultas y datos sobre nodos multicores, otras investigaciones proponen combinar procesamiento de consultas multihilo totalmente asíncronas con masivamente síncronas en base al nivel de tráfico de consultas.

Al día de hoy la mayoría del trabajo realizado se lleva a cabo sobre plataformas clásicas de memoria distribuida y compartida (clusters y nodos con varios

núcleos por procesador). Sin embargo, existen pocos grupos desarrollando estudios en torno a la búsqueda por similitud en espacios métricos sobre plataformas basadas en GPUs. Algunas soluciones generalistas que utilizan GPUs sólo abordan el problemas de consultas *kNN* sin utilizar estructuras de datos. En general, las GPUs básicamente se utilizan para paralelizar búsquedas exhaustivas por lo que no se utilizan estructuras métricas [10,11].

Por ejemplo, en [10] se propone dividir la base de datos de elementos (A) y la de consultas (B) en submatrices de tamaño fijo. De esta forma, la matriz resultante C de distancias es resuelta por bloques de hilos. Cada submatriz de distancias se ordena usando el *CUDA-based Radix Sort* [12] para posteriormente seleccionar los k primeros elementos como resultado final.

En [11] se implementa el algoritmo de fuerza bruta y se propone que cada hilo resuelva la distancia de un elemento de la base de datos contra la consulta, para luego ordenar el arreglo resultante con una variante del *insertion sort*.

En general, en los trabajos anteriores la paralelización es aplicada en dos etapas. La primera consiste en construir la matriz de distancias, y la segunda en el proceso de ordenamiento para obtener los resultados.

En [13] se presenta una variante de lo anterior. Este trabajo compara 2 estrategias, la primera, al estilo de los trabajos anteriores. Sin embargo, la segunda estrategia, llamada *Heap Based Reduction*, propone resolver una consulta por cada bloque. Después de haber calculado todas las distancias para una consulta (exhaustivamente), envía en cada lanzamiento de kernel un solo bloque, manteniendo un heap por cada hilo del bloque. Cada heap de tamaño k se utiliza para almacenar los k vecinos más cercanos a partir de las distancias entre los elementos de la base de datos y la consulta.

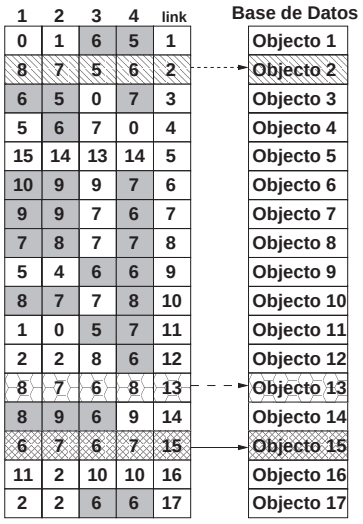
En [13,14] se utilizan estructuras métricas sobre una GPU y comparan sus resultados con versiones secuenciales. En [14] utiliza una estructura métrica y compara los resultados obtenidos para la búsqueda por rango contra versiones secuencial y multicore-CPU, mostrando una mejora notable al usar este nuevo tipo de plataforma.

2. Una Estructura Métrica Genérica Basada en Pivotes

En términos generales, las estructuras métricas Spaghettis [4], SSS-Index [6], LAESA [2], y otras, pueden ser consideradas como un arreglo bidimensional. Básicamente la diferencia entre éstas es la forma de obtener los pivotes y la forma de almacenar la tabla de distancias.

Una estructura métrica genérica (*Generic Metric Structure: GMS*) puede considerarse como una tabla de distancias entre los pivotes y los elementos de la base de datos. La construcción sería como la indicada en la Sección 1.1. La Figura 1(a) muestra un ejemplo de esta estructura.

La búsqueda por rango sobre esta estructura, dada una consulta q y un rango r, implicaría los siguientes pasos (algoritmo de la Figura 1(b)):



1	2	3	4	link	Base de Datos
0	1	6	5	1	Objecto 1
8	7	5	6	2	Objecto 2
6	5	0	7	3	Objecto 3
5	6	7	0	4	Objecto 4
15	14	13	14	5	Objecto 5
10	9	9	7	6	Objecto 6
9	9	7	6	7	Objecto 7
7	8	7	7	8	Objecto 8
5	4	6	6	9	Objecto 9
8	7	7	8	10	Objecto 10
1	0	5	7	11	Objecto 11
2	2	8	6	12	Objecto 12
8	7	6	8	13	Objecto 13
8	9	6	9	14	Objecto 14
6	7	6	7	15	Objecto 15
11	2	10	10	16	Objecto 16
2	2	6	6	17	Objecto 17

(a)

Generic Metric Structure : Algoritmo de Búsqueda.

rangesearch(query q, range r)

```
1 : {Sea 𝕐 ⊆ 𝕏 la base de datos}
2 : {Sea P un conjunto de pivotes p_1, ..., p_k ∈ 𝕏}
3 : {Sea D la tabla de distancias asociadas q}
4 : {Sea S el Índice Métrico}
5 : for all p_i ∈ P do
6 :     D_i ← d(q, p_i)
7 : end for
8 : for all y_i ∈ 𝕐 do
9 :     descartado ← false
10 :    for all p_j ∈ P do
11 :        if D_j − r > S_ij || D_j + r < S_ij then
12 :            descartado ← true
13 :            break;
14 :        end if
15 :    end for
16 :    if !descartado then
17 :        if d(y_i, q) ≤ r then
18 :            agrega y_i al resultado
19 :        end if
20 :    end if
21 : end for
```

(b)

Figura 1. Estructura métrica genérica y algoritmo de búsqueda. (a) Estructura. (b) Algoritmo de búsqueda por rango.

1. Para cada consulta q, se calcula la distancia entre q y todos los pivotes $p_1, \ldots, p_k$. Con esto se obtienen k intervalos de la forma $[a_1, b_1], ..., [a_k, b_k]$, donde $a_i = d(p_i, q) - r$ y $b_i = d(p_i, q) + r$.
2. Los objetos, representados en la estructura por sus distancias a los pivotes, son candidatos a la consulta q si todas sus distancias están dentro de todos los intervalos.
3. Para cada candidato y, se calcula la distancia $d(q, y)$, si $d(q, y) \leq r$, entonces el objeto y es solución a la consulta q.

La Figura 1(a) representa una estructura de datos métrica genérica (GMS). Esta estructura es construida usando 4 pivotes para indexar una base de datos de 17 objetos. La función de distancia es discreta. Para una consulta q con rango $r = 2$ y distancias $d(q, p_i) = \{8, 7, 4, 6\}$ define los intervalos $\{(6, 10), (5, 9), (2, 6), (4, 8)\}$. Las celdas marcadas en oscuro son aquellas que están dentro de los intervalos. Las celdas achuradas representan a aquellos objetos cuyas distancias están dentro de todos los intervalos y por lo tanto son candidatos. Finalmente, se calcula la distancia directa entre la consulta y los objetos candidatos $(2, 13, 15)$.

Para el presente trabajo se ha seleccionado una serie de estructuras que tienen características similares, incluida la versión genérica. Estas estructuras son:

Spaghettis: Es una estructura basada en pivotes y de tipo arreglo, no posee ningún método de selección de pivotes a priori. El índice es una tabla que contiene las distancias entre los elementos de la base de datos y los pivotes. Posteriormente, la tabla se ordena por distancia a cada uno de los pivotes, de esta

manera se puede obtener una reducción del tiempo de búsqueda al aplicar una búsqueda binaria previa para encontrar la primera distancia desde donde comenzará la búsqueda. La implementación utilizada en este trabajo es una versión ordenada sólo por el primer pivote.

SSS-Index: Esta estructura selecciona un conjunto dinámico de pivotes o centros utilizando una técnica denominada *Sparse Spatial Selection*. Sea $(\mathbb{X}, d)$ un espacio métrico, $\mathbb{Y} \subset \mathbb{X}$ y M la distancia máxima entre los dos objetos más alejados, $M = max\{d(x, y)/x, y \in \mathbb{Y}\}$. El primer pivote se selecciona arbitrariamente. Después, para cada elemento $y_i \in \mathbb{Y}$, y_i se selecciona como pivote sí y sólo sí está a una distancia mayor o igual a $M * \alpha$ de todos los pivotes ya seleccionados, siendo α una constante. Es decir, un objeto de la base de datos se toma como pivote si está situado a más de una fracción de la distancia máxima de todos los pivotes anteriores. En [6] se determinó que los valores óptimos para α están en torno a 0.4. Sin embargo, el valor del α depende mucho del tipo de espacio sobre el que se relizan los cálculos y de los rangos de búsqueda. En general el valor α no garantiza la elección de los mejores pivotes en todos los espacios.

MSD-Laesa: Linear Aproximating Search Algorithm es una estructura que usa un método de selección de pivotes conocido como *Suma Máxima de Distancias* (*Maximun Sum of Distances MSD*). La idea de este método es seleccionar un conjunto de pivotes definido que se encuentren lo más separados entre ellos. Se comienza con un pivote base seleccionado arbitrariamente. Los siguientes pivotes se seleccionan si estos están a la mayor distancia acumulada de todos los anteriores.

En los experimentos presentados en la siguiente sección, al comparar cantidades bajas y altas de pivotes se puede obtener una visión más amplia del comportamiento de las estructuras y extraer información útil al momento de su implementación en GPUs. Las estructuras se evalúan en las mismas condiciones.

3. Plataforma, Casos de Estudio y Primeros Resultados

La presente sección describe la plataforma utilizada así como también los casos seleccionados para el estudio de las distintas estructuras. También se presentan resultados preliminares necesarios para las etapas siguientes del artículo.

3.1. Ambiente Experimental

Para los experimentos se seleccionaron dos espacios métricos considerados representativos para este tipo de problemas. El primero corresponde a un diccionario español de 86.061 palabras. La distancia utilizada fue la *distancia de edición*, que corresponde al mínimo número de inserciones, eliminaciones o sustituciones necesarias para que una palabra sea igual a otra. Los rangos de búsqueda fueron de 1 a 4. El segundo espacio, de histogramas de colores, es un conjunto de 112.682 vectores (objetos de dimensión 112) de una base de datos de imágenes. Para este segundo espacio se ha elegido como función de distancia la distancia

Euclidiana y se usaron rangos que recuperaban el $0,01\,\%$, $0,1\,\%$ y $1\,\%$ desde el conjunto de datos. Ambas bases de datos se dividen en dos conjuntos aleatorios, el primero, de un $90\,\%$ de los datos se utiliza para construir la estructura y el $10\,\%$ restante se utiliza como consultas. Espacios de menor dimensión no fueron considerados dado que el interés es sobre espacios de dimensión mayor. Los costes de construcción no fueron considerados en los tiempos medidos.

Las características principales de la plataforma utilizada son:

- CPU: 2 Quadcore Xeon E5530 de 2.4GHz y 48GB de memoria principal.
- GPU: 4 Nvidia Tesla C1060 240 cores de 1.3GHz y 4 GB de memoria global.
- Herramientas de desarrollo: lenguaje C, OpenMP y SDK v3.2 de CUDA [8].

3.2. Resultados Preliminares

Para todas las estructuras, las cantidades de pivotes seleccionados fueron: $26, 44, 82, 328, 665, 1362$ para el espacio de palabras y $30, 35, 44, 57, 74, 119, 155, 244$ para el espacio de vectores. Estos valores corresponden a los obtenidos inicialmente al variar el parámetro α del método SSS. Para GMS, Spaghettis y Laesa se agregan 32 y 500 para palabras y 32 para vectores.

Las Figuras 2 y 3 muestran los resultados preliminares para las versiones secuenciales de los métodos seleccionados. Con el objetivo de poder visualizar mejor los resultados, los gráficos muestran sólo los intervalos importantes para las estructuras, entre 32 y 500 para palabras, y entre 32 y 119 para los histogramas de colores. En la Figura 2, y a efectos de claridad, se muestran en gráficas separadas los resultados correspondientes a los rangos 1 y 2 de los de 3 y 4.

En términos de evaluaciones de distancia, el comportamiento de las estructuras depende del número de pivotes y del rango. Es así como para los rangos menores se pude observar un punto de inflexión en torno a los 44 y 57 pivotes en ambos espacios. Para los rangos mayores, al aumentar el número de pivotes, las evaluaciones disminuyen. Sin embargo, en términos de tiempo de ejecución, a medida que crece el número de pivotes existe un cálculo adicional que hay que considerar, debido a comparaciones y desplazamientos dentro de la tabla de distancias.

En general, la estructura con mejor rendimiento en rangos bajos es la Spaghettis tanto para bajas como para altas cantidades de pivotes. Esto debido a que la búsqueda binaria realizada para encontrar el rango exacto para el primer pivote optimiza el proceso. La estructura GMS es la que le sigue en rendimiento. Es importante mencionar que la búsqueda en espacios métricos depende mucho de la forma del espacio, de la dimensión real y de la dimensión intrínseca, es por ello que muchos métodos no se comportan igual en todos los espacios. En particular, en los espacios elegidos, los métodos SSS y MSD no se comportaron como se esperaba, siendo la selección aleatoria de pivotes la de mejores prestaciones.

A medida que los rangos o radios aumentan, la búsqueda sobre la estructura Spaghettis se degrada. Esto es debido a que la cantidad de objetos que se descartan con el primer pivote, es mucho más bajo que para los rangos menores, con lo que en este caso la búsqueda binaria se transforma en un costo adicional.

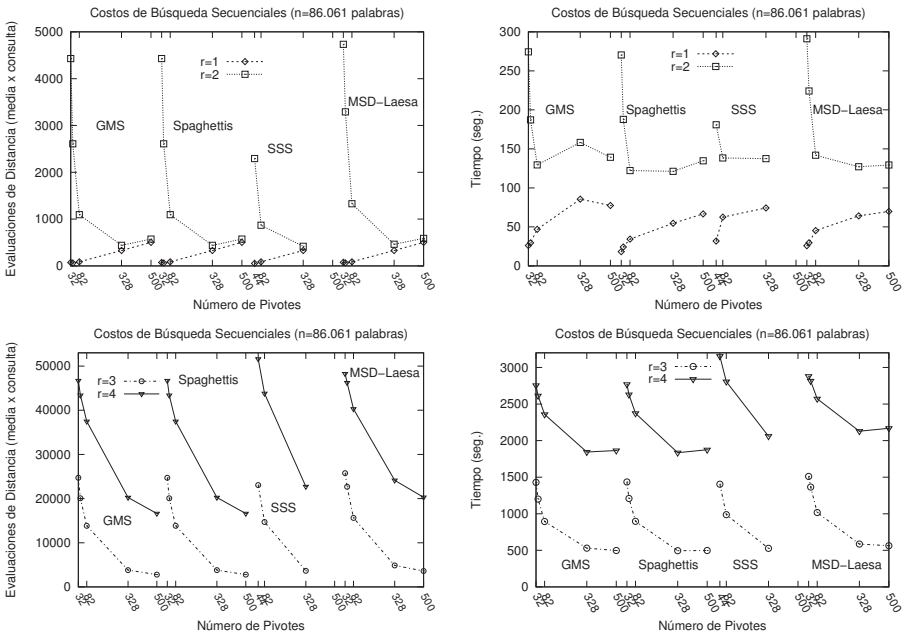


Figura 2. Cálculos de distancia promedio por consulta (columna izquierda) y tiempos totales de ejecución (columna derecha) para el espacio de palabras en español y las estructuras *GMS*, *SSS-Index*, *Spaghettis*, *MSD-Laesa*.

A modo de ejemplo, para el espacio de vectores y el mayor rango, los objetos descartados son menos del 50 % de la base de datos. Para la base de datos de palabras cuya dimensión intrínseca es alta, la cantidad de objetos descartados va desde los 36,5 % para el primer rango, a sólo el 3 % para el rango mayor (con un solo pivote).

4. Una Implementación Basada en GPU

La presente sección describe la implementación basada en GPU del algoritmo de búsqueda por rango para una estructura genérica (GMS).

Al implementar el algoritmo de búsqueda se consideró que cada consulta sería resuelta en forma completa e independiente. El algoritmo implementa las tres partes definidas en la Sección 2 y se resuelve en un kernel distinto cada una. Además, para optimizar el uso de los recursos de la GPU, se hace uso de la memoria compartida, tal que, en cada kernel se define un conjunto de datos que serán llevados a esta memoria.

1. Para optimizar el uso de la GPU, se resuelve en un único kernel la primera parte del algoritmo, es decir, se realiza el cálculo de las distancias entre todas las consultas Q y todos los pivotes P. El proceso consiste en lanzar el kernel

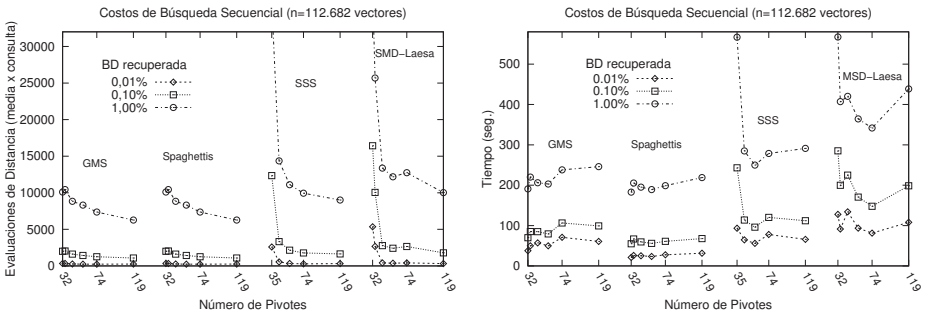


Figura 3. Cálculos de distancia promedio por consulta (izquierda) y tiempos totales de ejecución (derecha) para el espacio de histogramas de colores y las estructuras *GMS*, *SSS-Index*, *Spaghettis*, *MSD-Laesa*.

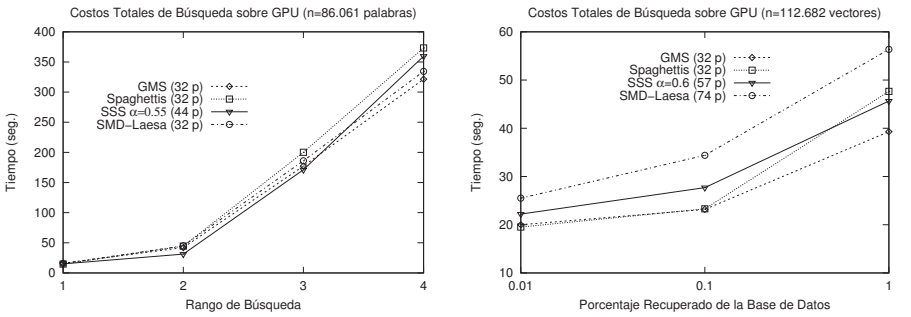
con tantos hilos como número de queries ($|Q|$), tal que, cada hilo resuelve de forma independiente las distancias de cada query a los pivotes. Como resultado, el kernel genera una matriz de tamaño $|Q| \times |P|$ con las distancias correspondientes.

En este caso, todos los hilos deben acceder a los pivotes, por lo que son estos los datos que se almacenan en memoria compartida, organizados de tal forma que los hilos accederán a las mismas posiciones de memoria.

2. La segunda parte de la paralelización se encarga de determinar si un elemento es o no candidato a una query. Para ello se lanza un segundo kernel que ejecuta un hilo por cada elemento (y_i) en la base de datos y determina si dicho dato es o no candidato, es decir, este kernel entrega la lista de candidatos para una query. En este caso, los datos comunes a todos los hilos que pueden ser llevados a memoria compartida son las distancias entre la query y los pivotes.
3. Finalmente, el último kernel se encarga de determinar cuál de los candidados obtenidos en la segunda parte son realmente solución a la consulta q. En este kernel, el número de hilos corresponde al número de candidatos por cada consulta y cada hilo resuelve si un candidato es o no solución. Al término del tercer kernel, se obtiene una lista con las soluciones para la consulta. El dato común a todos los hilos es la consulta, por lo que es este dato el llevado a memoria compartida.

De la Sección 3.2 se obtienen conclusiones que son utilizadas en esta fase del trabajo. De éstas, la más relevante es que para todas las estructuras y espacios coinciden en que a menor rango la mejor cantidad de pivotes es 32. Para los rangos mayores, las estructuras de mejor rendimiento siguen siendo Spaghettis y GMS, pero para el espacio de palabras los mejores valores están en torno a los 328 y 500, sin embargo, para el espacio de vectores el valor de 32 se mantiene.

Finalmente, dada las consideraciones anteriores, se eligió como número de pivotes para la estructura genérica la cantidad de 32. La Figura 4 muestra la



(a) Diccionario de palabras en español. (b) Histogramas de colores.

Figura 4. Tiempo de ejecución para GPU de las mejores versiones secuenciales.

comparación entre las estructuras sobre una plataforma con una GPU. Para todas las estructuras se eligieron las mejores versiones, en cantidades de pivotes, para el rango más bajo en cada espacio. Como sucedió en las veriones secuenciales, para rangos 3 y 4 en el espacio de palabras las estructuras con pivotes cercanos a 500 tuvieron un mejor rendimiento que las de 32 pivotes (no graficadas aquí). Para la implementación del Spaghettis en GPU, se obtuvo con búsqueda binaria el primer objeto dentro de los intervalos de la consulta y su posición fue pasada al kernel, de esta manera, los hilos anteriores a esta posición no son activados.

Se puede observar que el Spaghettis sólo obtiene el mejor rendimiento para el espacio de histogramas de colores y en el rango más pequeño. En ambos espacios esta estructura comienza a bajar el rendimiento a partir del segundo rango, lo que no ocurrió de manera tan notoria en la versión secuencial.

La estructura SSS-Index sólo obtuvo el mejor rendimiento para el espacio de palabras en los rangos 2 y 3. Para el primer kernel, tanto para SSS-Index como MSD-Laesa en el espacio de vectores, sus respectivas cantidades de pivotes no entraron completamente en la memoria compartida de la GPU. Esto implicó necesariamente un aumento en el tiempo de procesamiento.

Finalmente, los autores consideran que la estructura GMS es la que se comporta mejor sobre una plataforma basada en GPU, siendo la estructura que mantuvo un rendimiento más estable es todos los espacios.

5. Ventajas de una Plataforma Multi-GPU

Inicialmente se tienen dos alternativas para una implementación sobre una plataforma Multi-GPU. La primera implica distribuir la estructura métrica y la base de datos de tal manera que los núcleos o cores procesen sólo parte de cada una. Para este caso las consultas son replicadas sobre todos los cores. Sin embargo, esta alternativa requiere necesariamente una operación de reducción al finalizar todos los cores para reunir todas las soluciones de una consulta. Esta

solución es más adecuada para casos donde la estructura no entra completamente en la memoria de la GPU.

La segunda alternativa, que es la presentada en este trabajo, evita el coste de esperar y reunir las respuestas generadas por cada core. En este caso lo que se distribuye son las consultas y se replica la estructura métrica y la base de datos. Es decir, una GPU atiende completamente una consulta.

Las Figuras 5(a) y 5(b) muestran los resultados para una plataforma con 1 a 4 GPUs. Las Figuras 5(c) y 5(d) muestran los speed-up entre 2, 3 y 4 GPUs versus 1 GPU. Las figuras 5(e) y 5(f) muestran los speed-up entre la estructura genérica sobre las distintas cantidades de GPUs versus la mejor versión secuencial, en este caso, el Spaghettis de 32.

Las mejoras obtenidas con multiples GPUs sobre la mejor versión secuencial van de $4,7$ a un $11,5$ veces en el espacio de vectores y entre un $8,6$ y $32,6$ en el espacio de palabras. Para el speed-up calculado en base a una GPU, el aumento va de entre un $1,55$ a $2,47$ para los histogramas de colores y de un $1,92$ a un $3,76$ para las palabras en español.

6. Conclusiones y Trabajo Futuro

Este trabajo presenta una implementación de una estructura métrica genérica basada en pivotes para búsquedas por similitud en espacios métricos sobre una plataforma basada en GPU y multi-GPU.

De los resultados para las versiones secuenciales se pudo determinar que la estructura GMS tuvo un comportamiento similar al Spaghettis. En relación a las limitaciones de memoria compartida de la GPU, estos experimentos y los realizados sobre la GPU permitieron determinar el mejor número de pivotes para una óptima utilización de esta memoria. Para el espacio de vectores este valor fue de 32, con lo cual se logra que el primer kernel sea ejecutado completamente usando esta memoria. Para el espacio de palabras se determinó que 500 pivotes era la cantidad soportada por la memoria compartida, pero su rendimiento no era mejor que 32 para los rangos más bajos.

En las versiones para GPU, la estructura de mejor rendimiento fue la GMS, con mejoras sobre la mejor versión secuencial de hasta $8,6$ y $4,65$ veces para los espacios de palabras e histogramas de colores, respectivamente.

Finalmente, la versión multi-GPU desarrollada permite obtener rendimientos de hasta $11,5$ y $32,6$ veces más rápidos que la mejor versión secuencial para el espacio de vectores y el espacio de palabras respectivamente.

Como parte del trabajo futuro, queda pendiente realizar un análisis detallado de las diferentes estructuras examinadas en este artículo sobre distintos espacios métricos, de tal manera de obtener una visión más amplia de su comportamiento y de las ventajas que podrian implicar sobre una plataforma basada en GPUs.

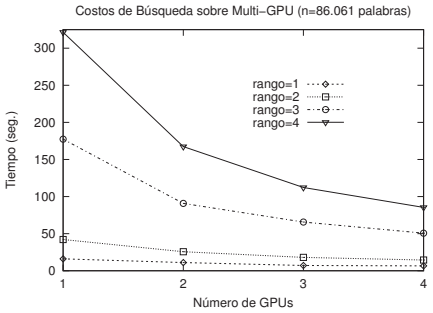
En particular, es importante realizar a futuro un análisis de las distintas formas de distribuir los datos sobre la plataforma multi-GPU y analizar diferentes ideas para minimizar las transferencias de datos entre la memoria utilizada por la CPU y la memoria global de la GPU.

Agradecimientos

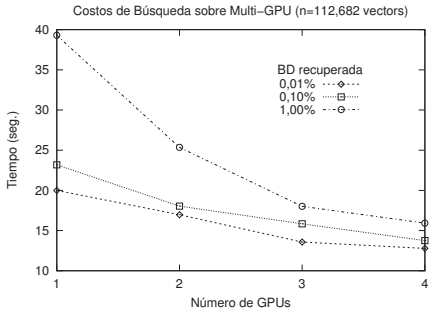
Este trabajo ha sido parcialmente financiado por el Ministerio de Ciencia e Innovación, proyecto SATSIM (Ref: CGL2010-20787-C02-02), España y la Universidad Nacional de la Patagonia Austral, proyecto 29/A274-1, Argentina.

Referencias

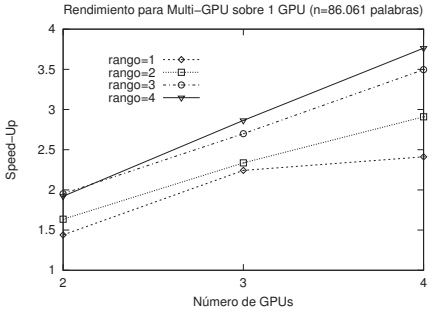
1. Chávez, E., Navarro, G., Baeza-Yates, R., Marroquín, J.L.: Searching in metric spaces. In: ACM Computing Surveys. (2001) 33(3):273–321
2. Micó, L., Oncina, J., Vidal, E.: A new version of the nearest-neighbor approximating and eliminating search (aesa) with linear preprocessing-time and memory requirements. Pattern Recognition Letters **15** (1994) 9–17
3. Baeza-Yates, R., Cunto, W., Manber, U., Wu, S.: Proximity matching using fixedqueries trees. In: 5th Combinatorial Pattern Matching (CPM'94). LNCS 807 (1994) 198–212
4. Chávez, E., Marroquín, J., Baeza-Yates, R.: Spaghettis: An array based algorithm for similarity queries in metric spaces. In: 6th International Symposium on String Processing and Information Retrieval (SPIRE'99), IEEE CS Press (1999) 38–46
5. Chávez, E., Marroquín, J., Navarro, G.: Fixed queries array: A fast and economical data structure for proximity searching. Multimedia Tools and Applications **14**(2) (2001) 113–135
6. Pedreira, O., Brisaboa, N.R.: Spatial selection of sparse pivots for similarity search in metric spaces. In: 33rd Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2007). Volume 4362 of LNCS., Harrachov, Czech Republic, Springer (2007) 434–445
7. Wu-Feng, Manocha, D.: High-performance computing using accelerators. Parallel Computing **33** (2007) 645–647
8. : NVIDIA CUDA C Programming Guide, Version 4.0. NVIDIA (2011) http://developer.nvidia.com/object/gpucomputing.html.
9. Grama, A., Karypis, G., Kumar, V., Gupta, A.: Introduction to Parallel Computing (2nd Edition). 2 edn. Addison Wesley (2003)
10. Kuang, Q., Zhao, L.: A practical GPU based kNN algorithm. International Symposium on Computer Science and Computational Technology (ISCSCT) (2009) 151–155
11. Garcia, V., Debreuve, E., Barlaud, M.: Fast k nearest neighbor search using GPU. Computer Vision and Pattern Recognition Workshop **0** (2008) 1–6
12. Satish, N., Harris, M., Garland, M.: Designing efficient sorting algorithms for manycore GPUs. International Parallel and Distributed Processing Symposium **0** (2009) 1–10
13. Barrientos, R., Gómez, J., Tenllado, C., Prieto, M., Marin, M.: kNN query processing in metric spaces using gpus. In: 17th International European Conference on Parallel and Distributed Computing (Euro-Par 2011). Volume 6852 of LNCS., Bordeaux, France, Springer (2011) 380–392
14. Uribe-Paredes, R., Valero-Lara, P., Arias, E., Sanchez, J., Cazorla, D.: Similarity search implementations for multi-core and many-core processors. In: International Conference on High Performance Computing and Simulation (HPCS). (2011) 656–663



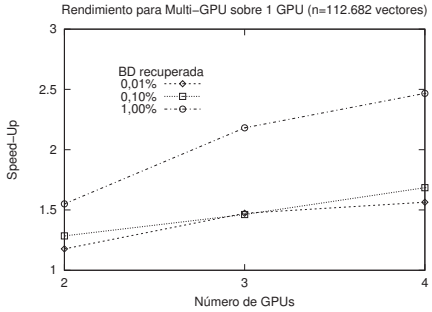
(a) Tiempos totales de búsqueda.



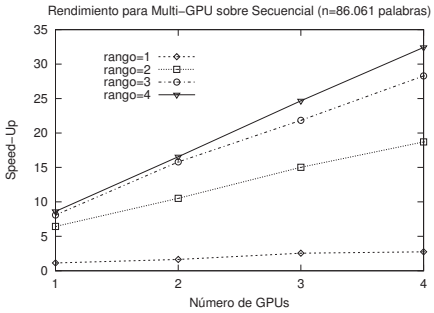
(b) Tiempos totales de búsqueda.



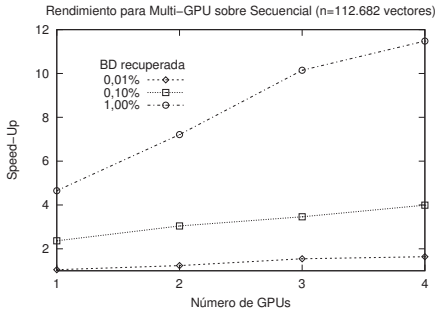
(c) 2, 3 y 4 GPUs versus una GPU.



(d) 2, 3 y 4 GPUs versus una GPU.



(e) Multi-GPU versus la mejor versión secuencial.



(f) Multi-GPU versus la mejor versión secuencial.

Figura 5. Resultados generales para la estructura métrica genérica (GMS) sobre una plataforma Multi-GPU, para el espacio de palabras (columna izquierda) y el espacio de histogramas de colores (columna derecha).

Practical Representations for Web and Social Graphs*

Francisco Claude[1] and Susana Ladra[2]

[1] University of Waterloo, Canada.
fclaude@cs.uwaterloo.ca
[2] Universidade da Coruña, Spain.
sladra@udc.es

1 Summary

Graphs are a natural way of modeling connections among Web pages in a network or people according to a criteria like friendship, co-authorship, etc. Many algorithms that compute and infer interesting facts out of these networks work directly over these graphs. Some examples of this are Connected components, HITS, PageRank, spam-detection, among others. In social networks, graph mining also enables the study of populations' behavior. Successful graph mining not only enables segmentation of users but also prediction of behavior. Link analysis and graph mining remains an area of high interest and development.

These human-generated graphs are growing at an amazing pace, and their representation in main memory, secondary memory, and distributed systems are getting more and more attention. Furthermore, space-efficient representations for these graphs have succeeded at exploiting regularities that are particular to the domain. In the case of Web graphs the main properties exploited are the locality of reference, skewed in/out-degree, and similarity of adjacency lists among nodes of the same domain. In social networks, there is a tendency towards forming cliques in the network.

Hence, representing, compressing and indexing graphs become crucial aspects for the performance and in-memory processing when mining information from those graphs. In this work we focus on obtaining space-efficient in-memory representations for both, Web graphs and social networks.

We first show how by just partitioning the graph and combining two existing techniques for Web graph compression, k^2-trees [Brisaboa, Ladra and Navarro, SPIRE 2009] and RePair-Graph [Claude and Navarro, TWEB 2010], exploiting the fact that most links are intra-domain, we obtain the best time/space trade-off for direct and reverse navigation when compared to the state of the art. Our proposal, which is called k^2-partitioned, splits the graph in $t + 1$ pieces, the first t ones correspond to sub-graphs formed by groups of domains. The last piece contains all the edges that point from one of the t sub-graphs to another one. This combination allows us to obtain the best cases for the k^2-tree, since most of the compression is gained inside domains and the query time is good when the matrix is dense. For the representation of the internal links we

* This work has been funded by NSERC, David R. Cheriton Scholarships program, Ministerio de Ciencia e Innovación (grants TIN2009-14560-C03-02 and CDTI CEN-20091048) and Xunta de Galicia (grant 2010/17).

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 81-82, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

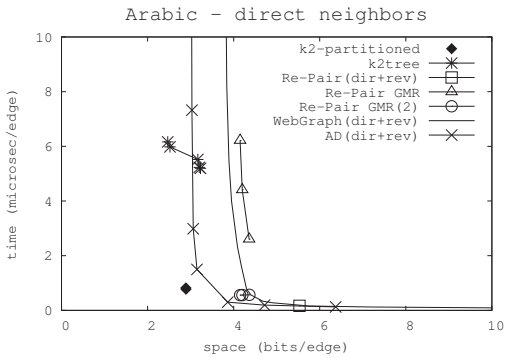


Fig. 1. Space/time trade-off to retrieve direct neighbors for several Web graphs.

use the k^2-tree and we represent external links using the approximate version of the Re-Pair compression method.

The experimental results show that for Web graphs we obtain a structure that improves upon the state of the art, achieving the best trade-off when we require both, direct and reverse navigation. This can bee seen in Figure 1, which shows the space/time trade-off for retrieving direct neighbors the large Web graph `arabic-2005` from the *WebGraph* project (reverse neighbors behave similarly). We measure the average time efficiency in μs/edge. We compare our compact representation, k^2-partitioned, with the original k^2-tree and other representations of the state of the art, including Re-Pair, Boldi and Vigna, and Apostolico and Drovandi's methods.

We can observe that our new representation k^2-partitioned competes successfully with the other compression methods of the literature, especially for larger graphs. It achieves very compact spaces, smaller than the rest of the techniques except for the k^2-tree, which can obtain slightly better spaces at the expense of degrading its time efficiency in orders of magnitude. Hence, our proposed technique achieves the best space/time trade-off for Web graph representation when direct and reverse navigation are required. Its simplicity contrasts with the remarkable results obtained.

We also study alternatives to compress social networks, where splitting the graph to achieve a good decomposition is not easy. For this case, we explore a new proposal for indexing MP_K linearizations [Maserrat and Pei, KDD 2010], which have proven to be an effective way of representing social networks in little space by exploiting common dense subgraphs. In the domain of social networks, our proposal improves upon previous results, both in theory and practice, showing that it constitutes a competitive index for social networks. Our implementations are available at `http://webgraphs.recoded.cl/`.

References

1. Claude F., Ladra S. Practical Representations for Web and Social Graphs. Proceedings of the 20th ACM International Conference on Information and Knowledge Management, CIKM'11. Glasgow, Scotland, UK. ACM Press, pages 1185–1190 (2011).

Reducción de la Complejidad Externa en Búsquedas por Similitud usando Técnicas de Clustering

Luis G. Ares, Nieves R. Brisaboa, Alberto Ordoñez, Oscar Pedreira*

Laboratorio de Bases de Datos, Universidade da Coruña
Campus de Elviña s/n, 15071, A Coruña, Spain
{lgares,brisaboa,alberto.ordonez,opedreira}@udc.es

Resumen La búsqueda por similitud tiene como finalidad determinar los objetos más semejantes o cercanos a uno dado. Los espacios métricos constituyen un modelo matemático que permite formalizar dicha búsqueda y que han dado lugar a diversos métodos, que tienen como objetivo principal reducir el número de evaluaciones de la función de distancia y el tamaño del índice. Las soluciones existentes son métodos basados en pivotes, que obtienen un número reducido de evaluaciones pero requieren cantidades importantes de espacio, y métodos basados en clustering, que necesitan poco espacio pero incrementan el número de evaluaciones. En este trabajo presentamos una nueva estrategia de clustering con sus algoritmos para búsquedas por rango y kNN que, reduciendo progresivamente el tamaño del cluster, disminuye significativamente la complejidad externa, un componente de la complejidad de los métodos existentes, con lo que se reduce el número de evaluaciones de la función de distancia.

Palabras Clave: Búsqueda por similitud, Espacios métricos, Reducción de cluster.

1 Introducción

El tratamiento de la información durante las últimas décadas ha abarcado toda clase de actividades humanas, originando tipos de datos complejos y un volumen de información en constante crecimiento. Entornos como los sistemas multimedia, la biología molecular y los sistemas de seguimiento y de recomendación, como los utilizados en las actividades industriales, financieras, sanitarias y sociales, ofrecen múltiples ejemplos de datos denominados *semiestructurados* y *no estructurados* donde los criterios de búsqueda no se basan en la exactitud, como ocurre con los tipos de datos tradicionales, y sí

* Trabajo parcialmente financiado por: Ministerio de Ciencia e Innovación (PGE y FEDER) refs. TIN2009-14560-C03-02, TIN2010-21246-C02-01, y ref. AP2010-6038 (programa FPU) para Alberto Ordóñez Pereira, y por Xunta de Galicia refs. 2010/17 (Fondos FEDER), y 10SIN028E.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 83-96, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

en la similitud, en el parecido o en la cercanía, lo que da origen a la denominada *búsqueda por similitud*.

El objetivo de la búsqueda por similitud es obtener los objetos más parecidos o más cercanos a uno dado, denominado *objeto de búsqueda* u *objeto a consultar*, mientras que una *búsqueda* o *consulta* se expresa mediante un objeto a consultar y un criterio de proximidad al mismo.

Los espacios métricos son un concepto matemático utilizado para representar alguna característica diferenciadora entre los elementos de un conjunto, mediante una función de distancia. Su aplicación como marco teórico para la resolución de la búsqueda por similitud sobre bases de datos ha dado lugar a diversos métodos. En general, dichos métodos realizan un preprocesado de los datos, creando índices que luego utilizan en el momento de realizar las búsquedas para, junto a las propiedades de los espacios métricos, intentar resolver las consultas sin comparar el objeto de búsqueda con cada elemento de la base de datos, sino con una pequeña parte, ya que el cálculo de las distancias es muy costoso [1–3].

Los tipos de consultas por similitud que se presentan con más frecuencia son la *búsqueda por rango*, $R(q, r)$, que determina los objetos que se encuentran a una distancia de q menor o igual que r, y la *búsqueda de los k-vecinos más cercanos*, $kNN(q)$, que obtiene los k elementos más cercanos a q.

Los métodos de acceso en espacios métricos pueden clasificarse en dos grupos, los basados en pivotes y los basados en clustering [1].

Los *métodos basados en pivotes* seleccionan un subconjunto de objetos de la colección, a los que denominan *pivotes*, y almacenan en el índice las distancias de los pivotes al resto de los objetos. Al realizar una búsqueda, comparan el objeto de búsqueda q con los pivotes y utilizan estas distancias para descartar el mayor número posible de objetos sin compararlos con q.

Los *métodos basados en clustering* particionan el espacio en conjuntos denominados *clusters*, en función de los objetos relevantes elegidos, denominados *centros*, y de sus distancias entre ellos, si la partición es mediante hiperplanos, o de la distancia entre cada centro y su objeto más alejado del cluster, denominada *radio de cobertura*, si la partición es por zonas esféricas. Cada objeto de la colección pertenece al cluster que corresponde a su centro más cercano. Dada una consulta, se compara el objeto de búsqueda q con los centros de cada cluster, de forma que se puede descartar la totalidad de un cluster, sin comparar sus objetos con q, si los datos del índice garantizan que ninguno de sus elementos cumple el criterio de búsqueda. Si un cluster no puede descartarse, se calcula la distancia de cada uno de sus elementos a q, en un proceso de exploración exhaustiva. Se conoce como *complejidad externa* el número de comparaciones de la función de distancia que se realizan en este proceso de exploración de los clusters no descartados.

En este trabajo presentamos una nueva estrategia de clustering que evita la búsqueda exhaustiva dentro de los clusters no descartados, reduciendo así la complejidad externa. Se basa en considerar una serie de regiones en cada cluster, para lograr que, mediante un proceso de reducción del cluster desde las zonas más alejadas a las más cercanas al centro, la búsqueda se circunscriba a

algunas regiones determinadas, pudiendo descartar las restantes. Llamamos a esta técnica *Reducción de Cluster*. La definición de las regiones en los clusters hace necesario incorporar datos adicionales al índice, pero la complejidad de su espacio se mantiene en $O(n)$. Introducimos en este trabajo los algoritmos para las búsquedas por rango y kNN aplicando la nueva estrategia, realizamos su evaluación experimental utilizando colecciones reales y sintéticas, y estudiamos la relación existente entre los requerimientos de espacio del índice y la mejora del rendimiento de las búsquedas. Los resultados obtenidos muestran una mejora del coste de las búsquedas en todos los casos analizados, dependiendo su porcentaje de las colecciones y del número de regiones utilizadas, siendo, por ejemplo, de entre un 13 y un 23% si el incremento de espacio es del 9%.

El resto del documento está organizado como se detalla a continuación. La Sección 2 revisa brevemente los métodos más destacados basados en clustering. La Sección 3 presenta los algoritmos para las búsquedas por rango y kNN. La Sección 4 muestra los resultados de la evaluación experimental. Por último, la Sección 5 presenta las conclusiones de este trabajo y futuras líneas de actuación.

2 Antecedentes y Trabajos Relacionados

Un *espacio métrico* es un par (X, d) formado por un universo de objetos X y una *función de distancia* o *métrica* $d : X \times X \longrightarrow \mathbb{R}^+$ que cumple las propiedades de identidad ($\forall\ x, y \in X,\ d(x, y) = 0 \Leftrightarrow x = y$), simetría ($\forall\ x, y \in X,\ d(x, y) = d(y, x)$) y desigualdad triangular ($\forall\ x, y, z \in X,\ d(x, y) \leq d(x, z)+d(z, y)$). La base de datos o colección de objetos es un subconjunto finito $S \subseteq X$ de tamaño $|S| = n$.

Los métodos de acceso en espacios métricos basados en clustering que particionan el espacio usando zonas esféricas, originan un conjunto de clusters $\{C_1, ..., C_m\}$, y almacenan en el índice datos sobre los centros de cada cluster, $\{c_1, ..., c_m\}$, y sobre los radios de cobertura o distancias de cada centro al objeto más alejado de su cluster. El centro de cada cluster y su radio de cobertura definen un círculo (c_i, cr_i) que contiene a todos los elementos del cluster.

Una búsqueda por rango $R(q, r)$ define un círculo de búsqueda (q, r). Para resolver la consulta, el objeto de búsqueda q se compara con el centro de cada cluster, obteniedo $d(q, c_i)$. Usando el criterio de descarte basado en el radio de cobertura, un cluster C_i puede descartarse sin comparar sus objetos con q, si el círculo de búsqueda (q, r) no interseca al círculo que contiene a los elementos del cluster, (c_i, cr_i), esto es, si $d(q, c_i) - cr_i > r$.

En el caso de las búsquedas kNN se determina el cluster más prometedor, que es el más cercano a q que puede tener objetos que formen parte de sus k vecinos más cercanos, y se calculan las distancias de q a cada uno de sus elementos. Dado que $\forall x \in \mathcal{C}_i,\ d(q, x) \geq d(q, c_i) - cr_i$, ocurre que $d(q, c_i) - cr_i$ es una cota inferior de la distancia de q a cualquier objeto de C_i, por lo que se utiliza para determinar sobre qué cluster actuar primero, y también como mecanismo de parada si esa distancia es mayor que la existente a los k vecinos ya detectados.

La complejidad de la búsqueda viene dada por la suma de la complejidad interna, que es el número de evaluaciones de la función de distancia necesarias para comparar q con los centros de los clusters, más la complejidad externa, que es el número de evaluaciones de la función de distancia requeridas para comparar q con los objetos que no han sido descartados [1].

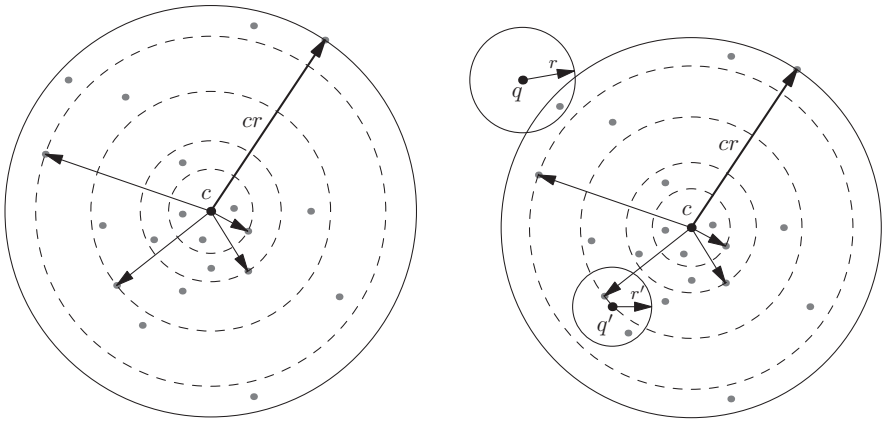
Los métodos basados en clustering superan a los basados en pivotes en que requieren menos espacio en sus índices y en su mejor comportamiento en espacios de grandes dimensiones, pero el número de evaluaciones de la función de distancia que originan es mayor. Los métodos de clustering existentes se diferencian fundamentalmente en los criterios de particionado del espacio y de descarte de los clusters, y en la estructuración y contenidos del índice. Para realizar el descarte de los clusters utilizan principalmente dos criterios, el basado en el radio de cobertura y el basado en hiperplanos.

BST [4] realiza un particionado recursivo usando siempre dos centros y creando un índice arbóreo, en el que almacena en cada nodo los dos centros y los radios de cobertura. GHT [5] usa el mismo criterio de particionado que el anterior, pero difiere en el método de descarte, de forma que siendo c_l y c_r los centros izquierdo y derecho respectivamente, se explora el subárbol izquierdo de cada nodo si $d(q, c_l) - r \leq d(q, c_r) + r$, y si ocurre que $d(q, c_r) - r \leq d(q, c_l) + r$ se explora el derecho. GNAT [6] extiende GHT al caso de más de dos centros por nodo, además de almacenar en cada nodo las distancias de los centros a los restantes clusters, lo que permite eliminar algunos de ellos sin compararlos con q. VT [7] realiza una mejora sobre BST usando dos o tres centros en cada nodo y almacenando en cada nodo el objeto más cercano del nodo padre. M-Tree [8] es un importante método con capacidades dinámicas y un buen rendimiento de las búsquedas, similar a GNAT en su construcción, pero en su criterio de descarte es más parecido a BST porque se basa en el radio de cobertura. SAT [9] realiza una aproximación al grafo de Delaunay utilizando un índice arbóreo y aprovecha la relación de proximidad entre los objetos, para reducir el número de funciones de distancia a calcular.

Los métodos anteriores usan un particionado recursivo y un índice arbóreo. Una propuesta diferente es Lista de Clusters [10], que particiona el espacio generando una lista jerarquizada de clusters. El proceso de construcción se inicia eligiendo el primer centro y determinando su radio de cobertura en función de un tamaño fijo o de un número de objetos determinado. En el proceso de búsqueda pueden descartarse todos los clusters restantes de la lista. Como en los anteriores métodos, si un cluster no se descarta, hay que explorarlo exhaustivamente.

3 Búsqueda por Similitud con Reducción de Cluster

En esta Sección presentamos la nueva estrategia incluyendo sus algoritmos de búsqueda por rango y kNN, y contrastándolos frente a Lista de Clusters, aunque nuestra estrategia puede aplicarse con el mismo planteamiento a los métodos de clustering que utilizan el radio de cobertura como criterio de descarte.



(a) Cluster dividido en regiones (b) Ejemplos de búsqueda por rango

Fig. 1. Búsqueda por rango con reducción de cluster.

El objetivo es reducir la complejidad externa evitando la búsqueda exhaustiva en los clusters que no pueden descartarse. Para ello se divide cada cluster en β regiones, determinadas por las distancias del centro a $\beta - 1$ objetos, de forma que estas distancias actúan como radios de cobertura interiores y se almacenan en el índice. Esta división permite realizar las búsquedas sobre cada una de las regiones, en un proceso de reducción del cluster desde las regiones más alejadas del centro, lo que conlleva que se puede descartar un conjunto de regiones, sin comparar sus objetos con el objeto de búsqueda, al garantizar que sus elementos no cumplen el criterio de búsqueda, de una manera similar a como se descartaban los clusters. La selección de las regiones se realiza intentando que todas ellas tengan el mismo número de elementos. Dado que la distribución de los objetos del cluster con respecto a su centro no es uniforme [10], si el número de objetos del cluster no es divisible por el número de regiones, en una de ellas se utiliza un número diferente de elementos.

La Figura 1(a) muestra un ejemplo en el que un cluster se divide en $\beta = 5$ regiones mediante la elección de $\beta - 1 = 4$ objetos, para los que se almacena su distancia al centro del cluster. Cada una de las regiones tiene cuatro objetos.

El número de regiones que pueden definirse va desde un mínimo de dos, hasta un máximo determinado por el número total de objetos del cluster, excluyendo el centro. Usando dos regiones se podría descartar la mitad de los objetos del cluster sin compararlos con el objeto de búsqueda; con todas las regiones se podrían descartar todos los objetos restantes a partir de uno dado. Cuantas más regiones se definen, mayor es la posibilidad de descartar más objetos, con lo que el número de evaluaciones de la función de distancia se reduce. Pero como en el índice se almacenan las distancias del centro a los objetos que determinan las

regiones, a medida que aumenta el número de regiones se incrementa el tamaño del índice, por lo que es necesario establecer un equilibrio entre el coste de la búsqueda y los requerimientos de espacio.

Hay otros métodos que realizan algún tipo de división de los clusters. *MVPT* [11] utiliza pivotes que crean zonas en los clusters del mismo nivel de un árbol. *PM-Tree* [12] consiste en incorporar a M-Tree un conjunto de pivotes que permiten dividir los clusters en regiones determinadas por esos pivotes. *M-Index* [13] realiza un particionado mediante hiperplanos haciendo corresponder a cada objeto un valor real que consta de una parte entera, que identifica al cluster, y de una parte decimal, que es la distancia al centro del cluster más cercano.

Las características más destacadas de Reducción de Cluster es que se trata de una estrategia que puede aplicarse a cualquier método de particionado que use el radio de cobertura como criterio de descarte y que las regiones se generan a partir del centro del cluster, realizando una división de los mismos que permite procesarlos más eficientemente en la realización de las búsquedas.

3.1 Búsqueda por rango

En la búsqueda por rango inicialmente se procede como en otros métodos de clustering existentes, esto es, dada una consulta $R(q, r)$, se calculan las distancias del objeto a buscar q a los diferentes centros de los clusters. Un centro de cluster y su radio de cobertura definen un círculo (c, cr), mientras que el objeto a buscar y el rango de la consulta determinan el círculo de búsqueda (q, r). Nuestro método se diferencia de los existentes en la manera de explorar un cluster que no resulta descartado. Consideramos los siguientes casos de interés:

a) Si $d(q, c) - cr > r$ ocurre que el círculo (c, cr) definido por el cluster no interseca al círculo de búsqueda (q, r), por lo que ninguno de sus objetos puede cumplir el criterio de búsqueda y la totalidad del cluster se descarta.
b) Si $d(q, c) - cr \leq r$ pero no ocurre que $d(q, c) + r \leq cr$, el círculo de búsqueda interseca al cluster, por lo que algunos de sus objetos pueden cumplir el criterio de búsqueda. En este caso, los métodos existentes que usan como criterio de descarte el radio de cobertura, exploran la totalidad del cluster comparando q con cada objeto del cluster. Nuestra estrategia realiza una división del cluster en regiones para realizar después una comparación también por regiones, empezando en las más alejadas del centro, de forma que pueden descartarse todas las regiones a partir de una dada, si la siguiente región no interseca al círculo de búsqueda.
c) Si $d(q, c) - cr \leq r$ y además $d(q, c) + r \leq cr$, el círculo de búsqueda está contenido en el cluster. La comprobación de las distancias de q a los objetos de las regiones, se ciñe a las regiones que intersecan al círculo de búsqueda.

Tanto en b) como en c) se reduce el número de evaluaciones de la función de distancia en la exploración de los clusters que no pueden descartarse, porque se pueden descartar regiones enteras dentro de cada cluster.

La Figura 1 refleja dos ejemplos de búsqueda por rango con Reducción de Cluster. Su lado izquierdo muestra un cluster de centro c y radio de cobertura

cr, con veinte objetos, que se ha dividido en cinco regiones determinadas por las distancias del centro a los cuatro objetos interiores, representadas mediante flechas de trazo más fino. En el lado derecho se representan dos búsquedas por rango. La primera, $R(q, r)$, es un ejemplo del segundo caso considerado, ya que el círculo de búsqueda interseca a la región más alejada del centro, pudiendo descartarse el resto de regiones y reduciendo las evaluaciones de la función de distancia a solo un 20% de los objetos del cluster. La segunda consulta, $R(q', r')$, es un ejemplo del tercer caso, y en ella solo hay que comparar q con los objetos de las regiones segunda y tercera, consideradas en dirección al centro del cluster.

3.2 Búsqueda *k*NN

Los métodos de clustering que usan como criterio de descarte el radio de cobertura, resuelven en general la búsqueda $kNN(q)$ calculando primero la distancia de q a los centros del cluster, determinando después el cluster más cercano al objeto de búsqueda, denominado cluster más prometedor, para comprobar las distancias de q a cada objeto del mismo, creando así una serie de candidatos a k-vecinos más cercanos y reiterando el proceso para los siguientes clusters más cercanos, hasta que los restantes clusters están a mayor distancia que el candidato a k-ésimo vecino más cercano.

En nuestra estrategia no se exploran todos los objetos del cluster más prometedor, sino que únicamente se comprueba la distancia de q a los objetos que pertenecen a su región más cercana al objeto de búsqueda. Una vez explorada esa región, se elimina del cluster, provocando una reducción del mismo, y se busca de nuevo el siguiente cluster más prometedor, incluyendo ahora el cluster reducido.

Para determinar cuál es el cluster más prometedor, se tiene que, dado un cluster $\mathcal{C}_i$, la expresión $d(q, c_i) - cr_i$ es una cota inferior para la distancia de q a cualquier objeto de $\mathcal{C}_i$, esto es, $\forall x \in \mathcal{C}_i,\ d(q, x) \geq d(q, c_i) - cr_i$, por lo que dados dos clusters $\mathcal{C}_i$ y $\mathcal{C}_j$, el cluster $\mathcal{C}_i$ es más prometedor que $\mathcal{C}_j$ si:

$$d(q, c_i) - cr_i < d(q, c_j) - cr_j \tag{1}$$

La Figura 2 muestra un ejemplo de búsqueda $2NN(q)$ con dos clusters. Primero se calculan las distancias de q a los centros de los clusters, de forma que los dos centros pasan a formar parte de la lista de candidatos a 2-vecinos más cercanos. Dado que la cota inferior de las distancias de q a cualquier objeto de $\mathcal{C}_1$, representada por d_1, es menor que la correspondiente al cluster $\mathcal{C}_2$, representada por d_2, se tiene que el cluster $\mathcal{C}_1$ es el más prometedor. Nuestra estrategia no compara q con la totalidad de los objetos de $\mathcal{C}_1$, sino solo con los de la región de $\mathcal{C}_1$ más cercana a q. Fruto de esa exploración, se actualiza la lista de candidatos a vecinos más cercanos, desplazando los dos objetos de la región a los centros de los clusters. A continuación se reduce el cluster $\mathcal{C}_1$ eliminando la región que ya ha sido explorada, pasando a ser la nueva cota inferior de las distancias de q a cualquier objeto del cluster reducido, la distancia representada por d_3.

El proceso prosigue determinando el nuevo cluster más prometedor, y como d_2 es menor que d_3, resulta ser $\mathcal{C}_2$, por lo que se repite el procedimiento de

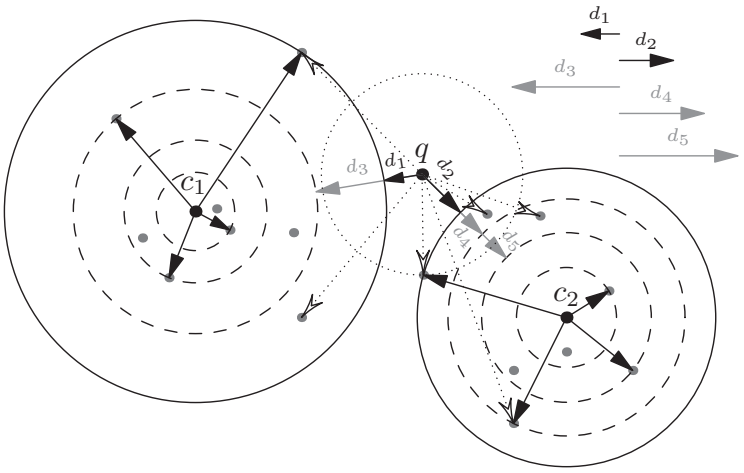


Fig. 2. Búsqueda kNN con reducción de cluster.

exploración de su región más alejada del centro pasando sus dos objetos a ser los nuevos candidatos a vecinos más cercanos y reduciendo el cluster al eliminar esa región. El siguiente cluster más prometedor lo determina que d_4 es menor que d_3, por lo que se explora la segunda región de $\mathcal{C}_2$, pero los objetos que se encuentran en ella no modifican la lista de candidatos porque están más alejados de q que los candidatos actuales. Las siguientes regiones a explorar las determinan d_3 y d_5, pero al ser mayores que la distancia de q al segundo candidato a vecino más cercano, que está representada en la figura por el círculo discontinuo, se garantiza que no se necesita explorar ninguna región más y la búsqueda finaliza, después de comprobar las distancias únicamente sobre tres regiones de las ocho posibles.

4 Análisis de Resultados

Presentamos los resultados obtenidos en la evaluación experimental de los nuevos algoritmos aplicados a Lista de Clusters, por ser representativo de la eficiencia de los métodos de clustering, bajo el marco propuesto por la librería *Metric Spaces Library*[1], utilizado seis colecciones de la librería, siendo unas reales y otras sintéticas, estando formadas unas por cadenas de caracteres y otras por vectores, e incluyendo una de gran dimensión. Las colecciones usadas son:

- English: una colección de 69.069 palabras de un diccionario de inglés.
- German: una colección de 75.086 palabras de un diccionario de alemán.
- Nasa: contiene 40.150 imágenes de los archivos de la NASA, representadas por vectores de dimensión 20.

[1] http://sisap.org/Metric_Space_Library.html

- Colors: contiene 112.544 histogramas de colores, representados por vectores de dimensión 112.
- Uniform-10, Uniform-12: colecciones de 100.000 vectores de dimensión 10 y 12 respectivamente, distribuidos uniformemente en el cubo unidad.

El 90% de las colecciones se ha utilizado como objetos a indexar, mientras que el 10% restante se ha destinado a consultas. En las colecciones de palabras los objetos se han comparado utilizando la distancia de edición y en las de vectores se ha usado la distancia euclídea. En las búsqueda por rango se ha utilizado $r = 2$ en las colecciones de palabras y en las de vectores, el que obtuviese el $0,01\%$ de la colección, en promedio, para cada consulta.

4.1 Búsqueda por rango

Para comprobar el rendimiento de la búsqueda por rango con Reducción de Cluster, hemos utilizado como tamaño de los clusters los valores de 10, 20, 40, 60, 80 y 100, y como valores de β o número de regiones definidas en cada cluster, los valores de 2, 5, 10 y 20, además de considerar también el número máximo de regiones, esto es, si cada objeto define una región.

La Figura 3 muestra el número medio de evaluaciones de la función de distancia frente al tamaño del cluster, considerando las diferentes colecciones y métodos. El método Lista de Clusters se ha representado en la figura con "`LC`". Los restantes métodos aplican Reducción de Cluster a Lista de Clusters, y se representan mediante "`LC CR-i`" siendo $i = \beta$ el número de regiones en las que se ha dividido el cluster, y con "`LC CR-all`" el caso de utilizar tantas regiones como objetos en el cluster.

Los resultados obtenidos muestran que aplicando Reducción de Cluster se obtiene una reducción significativa del número de evaluaciones de la función de distancia. Usando todos los objetos del cluster se obtiene el mejor resultado, pero con solo un número reducido de regiones, se obtiene ya una mejora sustancial, de forma que si se definen cinco regiones se obtienen unos resultados cercanos a los mejores posibles. Consideramos que este es un resultado destacable. Por otra parte, una vez que se tienen 10 o 20 regiones, incrementar su número contribuye de una manera muy reducida a mejorar el rendimiento.

4.2 Búsqueda *k*NN

La efectividad de las búsquedas kNN con Reducción de Cluster se ha contrastado para valores de k de 1 a 10, y para los mismos números de regiones utilizados en las consultas por rango, esto es, valores de β de 2, 5, 10, 20 y también el resultante de usar todos los objetos de los clusters. Este tipo de búsqueda se ha efectuado eligiendo un tamaño de cluster de 40, ya que este tamaño produjo buenos resultados para todas las colecciones en un contraste previo.

La Figura 4 refleja el número medio de evaluaciones de la función de distancia en función del número de vecinos, para las mismas colecciones y métodos que en el caso de las búsquedas por rango, por lo que se usan las mismas etiquetas.

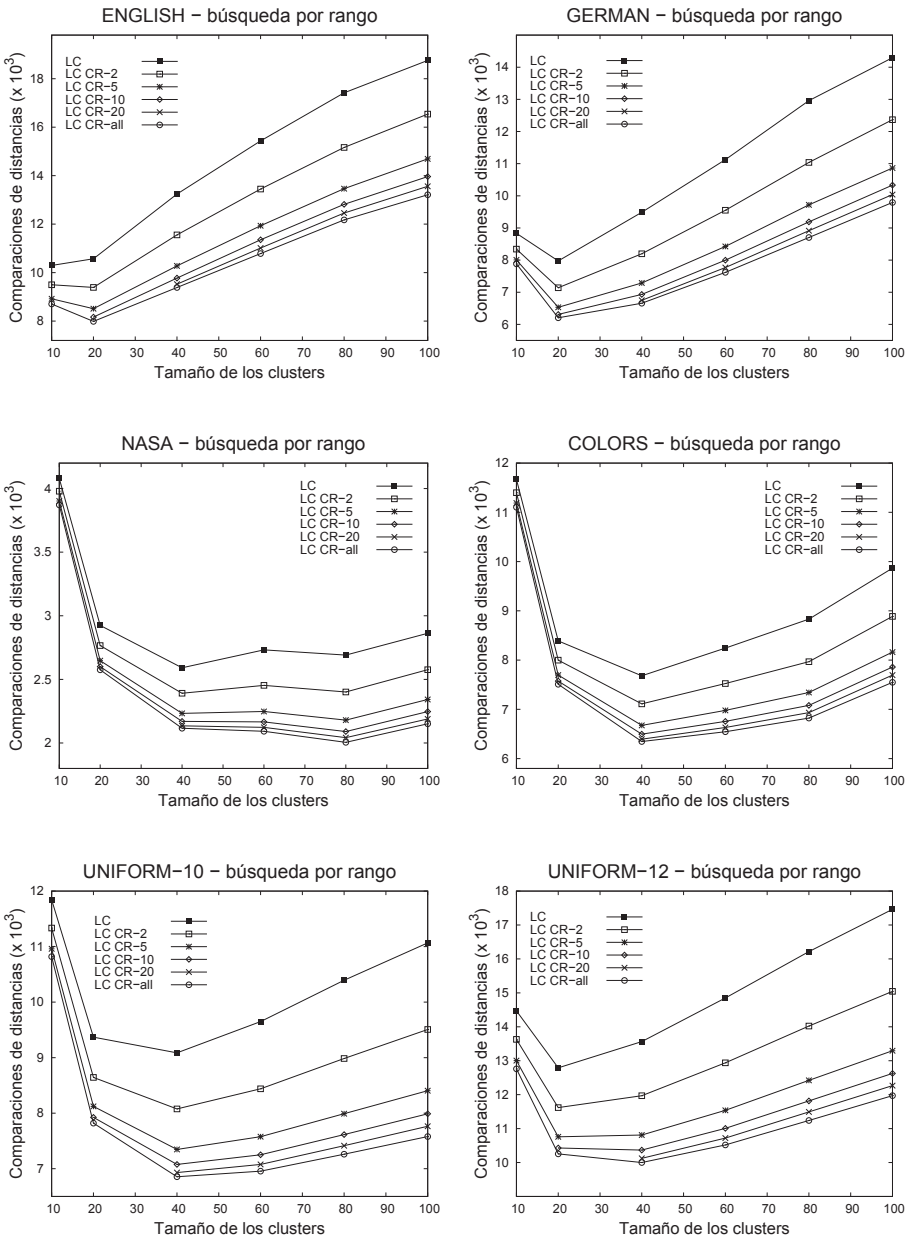


Fig. 3. Rendimiento de las búsquedas por rango. Cada figura muestra el número medio de evaluaciones de la función de distancia (en miles) en función del tamaño del cluster.

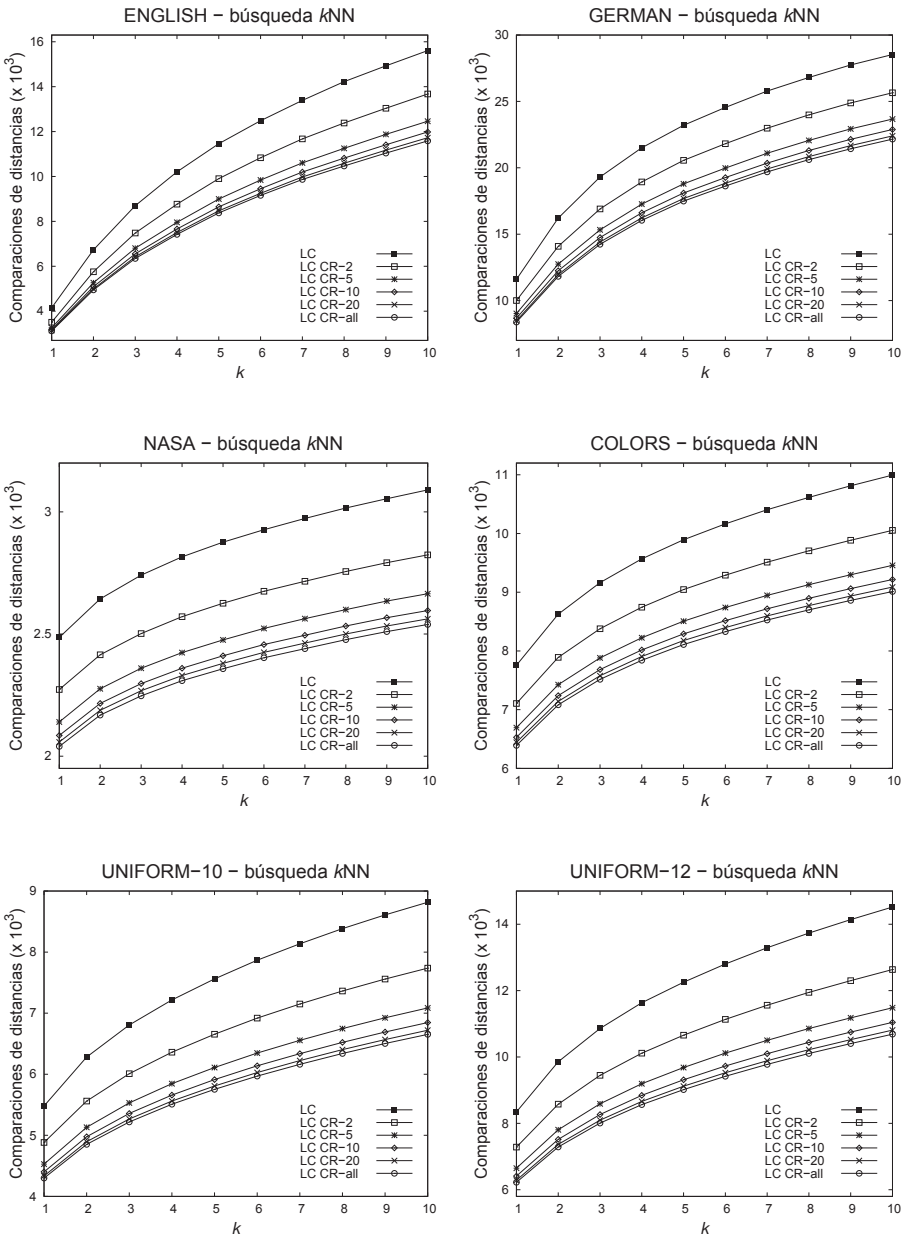


Fig. 4. Rendimiento de las búsquedas kNN. Cada figura refleja el número medio de evaluaciones de la función de distancia (en miles) en función del número de vecinos k.

Los resultados muestran un comportamiento similar al obtenido en las búsquedas por rango. Por ejemplo, con cinco regiones se obtienen ya resultados muy significativos, y a partir de diez, son muy cercanos a los mejores posibles. Los resultados son homogéneos para todos los valores de k en todas las colecciones.

4.3 Rendimiento de las Búsquedas y Requerimientos de Espacio

El mayor rendimiento de las búsquedas se logra al utilizar todos los objetos del cluster, excepto el centro, definiendo cada uno de ellos una región. Pero como en el índice se almacenan las distancias del centro a cada uno de los objetos utilizados para definir las regiones, en el caso de usar todos los objetos el espacio ocupado por el índice es también el más alto. De la misma forma, cuantas menos regiones se determinan, menor es el espacio necesario en el índice.

La Tabla 1 muestra el tamaño del índice, tanto al usar Lista de Clusters original, como al aplicarle Reducción de Cluster, para los diferentes números de regiones utilizadas en la experimentación. La columna "Relativo" indica el tamaño relativo del índice, también con respecto a Lista de Clusters original. Se aprecia un incremento del tamaño del índice de un 2% al considerar dos regiones, de un 9% al definir 5, y de un 20% si se consideran 10 regiones. A partir de este número, el tamaño del índice sufre un crecimiento considerable, llegando al 89% si se consideran todos los objetos del cluster. Estos porcentajes son idénticos en todas las colecciones.

La Figura 5 aporta una visión complementaria, ya que refleja la relación entre la mejora del coste de la búsqueda por rango y el incremento del espacio ocupado por el índice, para un tamaño de cluster igual a 40, expresando los resultados en valores relativos de porcentajes comparados con Lista de Clusters original. Los valores del eje de abscisas se corresponden con los de la última columna de la Tabla 1. Se muestran todas las colecciones utilizadas y el efecto de realizar las divisiones de los clusters en los números de regiones considerados, esto es, para los valores de β iguales a 2, 5, 10, 20 y también el obtenido al usar todos los objetos de los clusters. Por ejemplo, para la colección "German" al usar un $\beta = 2$, esto es, dos regiones, se genera un incremento del espacio del índice de un 2%, como se refleja en la Tabla 1, y se produce una mejora del coste de la búsqueda del 13,59%, obtenida en función de la reducción del número de evaluaciones de la función de distancia, ya que se pasa de 9.488,59 a 8.198,24, valores que se

Tabla 1. Efecto en los requerimientos de espacio (Kbytes).

β	English	German	Nasa	Colors	Relativo
LC	260,65	283,34	151,53	424,66	1,00
2	266,57	289,79	154,97	434,31	1,02
5	284,34	309,10	165,29	463,26	1,09
10	313,95	341,29	182,50	511,50	1,20
20	373,17	405,66	216,92	607,99	1,43
all	491,60	534,41	285,74	800,96	1,89

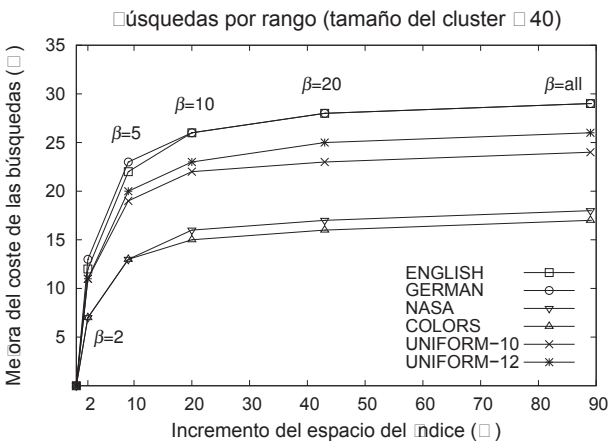


Fig. 5. Mejora del coste de la búsqueda por rango frente al incremento del espacio del índice, con tamaño del cluster igual a 40.

utilizaron en la subfigura superior derecha de la Figura 3, para el tamaño de cluster igual a 40, en las gráficas "LC" y "LC CR-2".

Se observa, por ejemplo, que con solo dos regiones, lo que supone un incremento de espacio de solo el 2%, la mejora de las búsquedas alcanza un 13% en la mayoría de las colecciones, y que definiendo cinco regiones ($\beta = 5$), se produce una mejora del rendimiento de la búsqueda que oscila entre un 13 y un 23%, a cambio de un incremento de espacio del índice del 9%. Al considerar 10 regiones, la mejora del coste de las búsquedas se incrementa muy suavemente, y a partir de este número, deja ya de ser significativa. Esto muestra que, eligiendo un número muy reducido de regiones, se obtienen mejoras importantes del coste de las búsquedas, independientemente del tamaño del cluster e incrementando el espacio del índice en un pequeño porcentaje.

5 Conclusiones

En este trabajo presentamos una nueva estrategia de clustering y sus algoritmos de búsquedas por rango y kNN, que evitan la comparación exhaustiva del objeto de búsqueda con los objetos de los clusters que no pueden descartarse con la información contenida en el índice, lo que supone la reducción de la complejidad externa. Hemos implementado los algoritmos para Lista de Clusters, pero son aplicables a cualquier método basado en clustering que usa como criterio de descarte el radio de cobertura.

Nuestra estrategia se basa en definir un número de regiones dentro de cada cluster, procurando que todas las regiones tengan el mismo número de objetos. Al realizar la búsqueda en un cluster no descartado, se procesa cada una de las

regiones, empezando por las más alejadas del centro, originando una reducción progresiva del cluster, hasta que pueden descartarse las regiones restantes. La definición de un reducido número de regiones dentro de cada cluster, reduce significativamente el coste de las búsquedas, originando un pequeño incremento del tamaño del índice y manteniendo la complejidad del espacio en $O(n)$.

Los resultados de la evaluación experimental, realizada sobre colecciones tanto reales como sintéticas pertenecientes a la *Metric Spaces Library*, muestran una mejora significativa del coste de las búsquedas, dependiendo de la colección y del número de regiones definidas, y el hecho de que si usamos un número reducido de regiones, la mejora es muy cercana a la que se obtiene si cada uno de los objetos del cluster define una región, que sería el caso de menor coste.

Como trabajo futuro estudiamos varias líneas de actuación, entre ellas la implementación de Reducción de Cluster a otros métodos que usan como criterio de descarte el radio de cobertura, su aplicación a otras operaciones como el join por similitud, y la evaluación de su escalabilidad en el tratamiento de grandes volúmenes de datos.

Referencias

1. Chávez, E., Navarro, G., Baeza-Yates, R., Marroquín, J.L.: Searching in metric spaces. ACM Computing Surveys **33** (2001) 273–321
2. Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity search. The metric space approach. Volume 32 of Advances in Database Systems. Springer (2006)
3. Hjaltason, G.R., Samet, H.: Index-driven similarity search in metric spaces. ACM Transactions on Database Systems **28**(4) (2003) 517–580
4. Kalantari, I., McDonald, G.: A data structure and an algorithm for the nearest point problem. IEEE Tran. on Software Engineering **9** (1983) 631–634 IEEE Press.
5. Uhlmann, J.K.: Satisfying general proximity/similarity queries with metric trees. Information Processing Letters **40** (1991) 175–179
6. Brin, S.: Near neighbor search in large metric spaces. In: Procs. of Conf. on Very Large Databases (VLDB'95), Morgan Kaufmann Publishers (1995) 574 – 584
7. Dehne, F., Noltemeier, H.: Voronoi trees and clustering problems. Information Systems **12**(2) (1987) 171–175
8. Ciaccia, P., Patella, M., Zezula, P.: M-tree: An efficient access method for similarity search in metric spaces. In: Procs. of Conf. on Very Large Databases (VLDB'97), ACM Press (1997) 426–435
9. Navarro, G.: Searching in metric spaces by spatial approximation. In: Procs. of String Processing and Information Retrieval (SPIRE'99), IEEE Press (1999) 141–148
10. Chávez, E., Navarro, G.: A compact space decomposition for effective metric indexing. Pattern Recognition Letters **26**(9) (2005) 1363–1376
11. Bozkaya, T., Ozsoyoglu, M.: Distance-based indexing for high-dimensional metric spaces. In: Proc. of the ACM International Conference on Management of Data (SIGMOD'97), Tucson, Arizona, USA, ACM Press (May 1997) 357–368
12. Skopal, T., Pokorný, J., Snásel, V.: Pm-tree: Pivoting metric tree for similarity search in multimedia databases. In: ADBIS (Local Proceedings). (2004)
13. Novak, D., Batko, M.: Metric index: An efficient and scalable solution for similarity search. In: Procs. of 2nd Int. Workshop on Similarity Search and Applications (SISAP'09), IEEE Press (2009) 65–73

NASS: A Semantic Annotation Tool for Media*

Angel L. Garrido[1], Oscar Gómez[1], Sergio Ilarri[2] and Eduardo Mena[2]

[1] Grupo Heraldo - Grupo La Información. Zaragoza - Pamplona, Spain.
{algarrido, ogomez}@heraldo.es
[2] IIS Department, University of Zaragoza, Zaragoza, Spain.
{silarri, emena}@unizar.es

Abstract. Nowadays media companies have serious difficulties for managing large amounts of news from agencies and self-made articles. Journalists and documentalists must face categorization tasks every day. There is also an additional difficulty due to the usual large size of the list of words in a thesaurus, which is the typical tool used to tag news in the media.
In this paper, we present a new method to tackle the problem of information extraction over a set of texts where the annotation must be composed by thesaurus elements. The method consists of applying lemmatization, obtaining keywords, and finally using a combination of Support Vector Machines (SVM), ontologies and heuristics to deduce appropriate tags for the annotation. We carried out a detailed evaluation of our method with a real set of changing news and we compared out tagging with the annotation performed by a real documentation department, obtaining really promising results.

Keywords: Semantic tagging and classification; Information Extraction; NLP; SVM; Ontologies; Text classification; Media; News.

1 Introduction

In almost every company in the media industry, activities related to categorization can be found: news production systems must filter and sort the news at their entry points, documentalists must classify all the news, and even journalists themselves need to organize the vast amount of information they receive. With the appearance of the Internet, mechanisms for automatic news classification often become indispensable in order to enable their inclusion in web pages and their distribution to mobile devices like phones and tablets.

To do this job, medium and big media companies have documentation departments. They label the news they can, and the typical way to do that is by using thesauri. A thesaurus [1] is a set of items (words or phrases) used to classify things. These items may be interrelated, and it has usually the structure of a hierarchical list of unique terms.

* This research work has been supported by the CICYT project TIN2010-21387-C02-02.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 97-108, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

We have worked with real data owned by publications of Spanish media companies associated to the Vocento[3] Media Group. These companies have a documentation department and they use *EMMA*[4] as their archive platform. News in EMMA are represented as records in a relational database. Every record includes the text of the article. These articles are tagged every day by the documentation department of these companies. The tagging consists of filling several fields in every record, and one of the most important is the thesaurus field. In that field, documentalists can write as many thesaurus terms as they want from EMMA's thesaurus hierarchical tree of terms. This is an assisted process, but anyway it takes much time because of the high number of news produced every day. It is a tedious work subject to a lot of human errors and it is very subjective, so it is very easy not to be rigorous. This is especially true when different people with different opinions are working together with a list of near one thousand thesaurus terms. Our goal is to help them do this daily and difficult job.

In this paper, we focus on the inner workings of the NASS system (*News Annotation Semantic System*), which provides a new method to obtain thesaurus tags using semantic tools and information extraction technologies. The seminal ideas of this project have been recently presented under a condensed format [2] and they have received a good acceptance. With this paper, we want to delve deeper into the operation of the system and to show the results in more detail.

In our system we first propose to obtain the main keywords from the article by using text mining techniques. At the same time, using Natural Language Processing (NLP) [3], the system retrieves other type of keywords called *named entities*. Second, NASS applies Support Vector Machines (SVM) [4] text classification in order to filter articles. Third, it uses the keywords and the named entities of the filtered texts to query an ontology about the topic these texts are talking about. Then, NASS uses the answers to those queries to increase the probability of obtaining correct thesaurus elements for each text, and it updates that matching score in a table. Finally, the system looks at this table and selects the terms with a score higher than a given threshold, and then it labels the text with the corresponding tags. We have tested this method over a set of thousands of real news published by a media company. As we had the chance to compare our results with the real tagging performed by the documentation departments, we have benefited from this real-world experience to evaluate our method [6].

This paper is structured as follows. Section 2 explains our method in detail. Section 3 discusses the results of our experiments with real data. Section 4 is a brief explanation of the state of art with SVM and Ontology-Based Information Retrieval. Finally, Section 5 provides our conclusions and some lines of future work.

[3] Vocento is a multimedia group in Spain, consisting of over 100 companies. For further information, please see *http://www.vocento.com/en*.

[4] EMMA is the Spanish abbreviation of *MultiMedia Environment Archiver*, which is a proprietary solution developed by Ibercentro Media.

2 NASS Methodology

In this section, we present our approach. First, we provide an overview of the architecture, and then we focus on the most interesting steps during the document annotation.

2.1 The System Architecture

Methods to obtain suitable tags for a text have been studied in the context of Information Extraction (IE). According to Russell and Norvig [7], IE means automatically retrieving certain type of information from natural language text. They say that IE is halfway between Information Retrieval (IR) systems and text understanding systems. However, according to a deep study of these issues recently performed by Wimalasuriya and Dou [5], works like ours could be classified in the field of Ontology-Based Information Extraction (OBIE), an emergent subfield of IE. Thus, it complies with the features identified in [5]:

- The system is going to process semi-structured natural language texts.
- The output is presented by using *ontologies*. An ontology is defined as a formal and explicit specification of a shared conceptualization [8], and consists of several components such as classes, data type properties, object properties, instances, property values of the instances, and constraints.
- The system uses an information extraction process guided by an ontology.

The main elements in an OBIE system are a preprocessor that works over the incoming text, an information extraction module (usually guided by a semantic lexicon like WordNet [9] and by a human-made ontology), and finally a Knowledge base used for storing the system's response. The architecture of our solution which is shown in Figure 1, fits in the general schema presented in [5] as seen later.

2.2 Implementation of the Proposal in a Real Context

In this section, we position our system in the production time line of a real media company. After finishing the everyday production, the news are obtained from the production system. Then, they are processed by EMMA and they are sent to a relational database. In that very moment the system has to produce tags, before documentalists access EMMA. There are some metadata already available, such as the date, section and author. Of course NASS also has the text and it is even able to identify different parts: the title, subtitle, introduction, signature, and other elements. However, there are several requirements to consider in the development of our system: the tags we obtain must belong to the documentation thesaurus, we have to use free software to develop our approach, and the tags should be obtained as fast as possible.

The outline of our method is as follows. First, NASS obtains from the text the names of the main characters, places and institutions, and then it deduces which

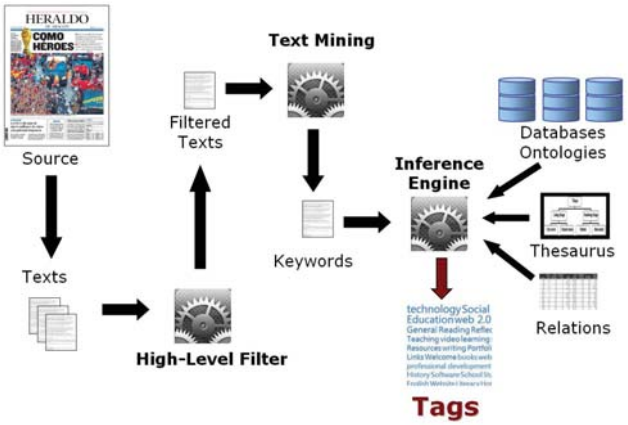


Fig. 1. General architecture of the NASS System.

the key ideas and major themes are. Second, the system uses this information in order to find related thesaurus terms. Finally, NASS assigns the thesaurus terms (with their ancestors) to the text. It looks easy, as basically it represents the way a person does this job. Unfortunately, making this task automatic is not that simple, as software does not understand text and lacks information about the context. So, we must look for some strategies that make it possible to obtain coherent terms from the thesaurus:

- The system has to obtain relevant keywords and named entities from the text, by using appropriate algorithms and a language analyzer.
- As soon as NASS obtains a list of keywords and named entities, it has to deduce their nature and find related thesaurus terms, by using SVM text categorization, a knowledge base, and a heuristic method.

Once the appropriate thesaurus terms have been identified, relating text with them and with their ancestors is trivial, as the system only has to traverse the thesaurus tree from the corresponding selected term upwards to the root. So, in the next subsections we will focus on the two first aforementioned tasks.

2.3 Lemmatization and Assignment of Keywords

Before obtaining keywords, NASS is going to lemmatize all the words in the text. So, lemmatization is the first step in the process, which means grouping together the inflected forms of a word. A lemma can be defined as the canonical form representing each word. This process simplifies the task of obtaining keywords and reduces the number of words the system has to consider later. It can also help to obviate *stop words*: prepositions, conjunctions, articles, numbers and other meaningless words. Moreover, it also provides us clues about the kinds

of words appearing in the text: nouns, adjectives, verbs, and so on. For this purpose, we have used Freeling [10]. Freeling is an open source suite of language analyzers developed at TALP Research Center, at BarcelonaTech (Polytechnic University of Catalunya). After the lemmatization process, the system has a list of significant words, which is the input to the next process: obtaining keywords.

We propose merging two methods to obtain a set of significant keywords from a given text. The first method that NASS applies is a simple term frequency algorithm [11], but with some improvements. We call it *TF-WP* (*Term Frequency – Word Position*), which is obtained by multiplying the frequency of a term with a position score that decreases as the term appears for the first time towards the end of the document. This heuristic is very useful for long documents, as more informative terms tend to appear towards the beginning of the document. The TF-WP keyword extraction formula is as follows:

$$TF - WP = (\frac{1}{2} + \frac{1}{2} * \frac{nrWords - pos}{nrWords}) * TF$$

$$TF = \frac{nrRepetitions}{nrWords}$$

where $nrWords$ is the total number of terms in the document, pos is the position of the first appearance of the term, TF is the frequency of each term in the document, and $nrRepetitions$ is the number of occurrences of that term.

The second method the system uses is the well-known *TF-IDF* (*Term Frequency – Inverse Document Frequency*) [12], based on the word frequency in the text but also taking into account the whole set of documents, not only the text considered. The TF-IDF keyword extraction formula is:

$$TF - IDF = TF * log_{10}(\frac{nrDocs}{D})$$

where TF is the frequency of each term in the text (as in TF-WP), $nrDocs$ is the total number of documents, and D is the number of texts that contain that word. We have merged both methods by adding the two values TF-WP and TF-IDF after applying them a weight α and β, respectively. At the end of this task NASS obtains a list of keywords with their number of repetitions and weights.

2.4 Identification of Named Entities

In news, it is very common to find names of people, places or companies. These names are usually called *named entities* [13], which share a common feature: they start in uppercase. For example, if the text contains the words "Dalai Lama" both words could be considered as a single named entity to capture its actual meaning. This is a better option than considering the words "Dalai" and "Lama" independently. To do this task we have used Freeling too, which uses this identification method and provides a confidence threshold to decide whether

to accept a named entity or not. In our system, this threshold is set at 75% in order to ensure a good result. NASS retrieves the more relevant named entities identified, replaces whitespaces by underscores (for example, "Dalai_Lama"), and finally adds them to the same collection of keywords obtained before.

2.5 Text Categorization

At this point, NASS has a list of the most important keywords and named entities in the input article. It could tag the text with this information, but we want to take a step forward by using the thesaurus for tagging. The question now is: how could NASS obtain thesaurus terms from a list of keywords? Performing a simple text comparison is not enough. For example, "SOUTH_AMERICA" is a thesaurus term used in documentation departments, but if an article is about *Argentina* it is unlikely that the words "South America" will appear. That article could speak about places or people in Argentina, and the keywords could be for instance *Argentina, Buenos_Aires, Cristina_Fernandez*. With that information, the system has to deduce that the thesaurus term desired is only "SOUTH_AMERICA". So, to be able to label that article with the correct thesaurus term, the system must know that Argentina is a country that belongs to South America or that Cristina Fernandez is the president of Argentina.

There are a lot of interesting ways to infer and deduce terms according to their meaning and their relationships with other terms. We have chosen SVM text categorization because it is a powerful and reliable tool for text categorization [4]. Regarding the type of SVM used, we have used a modified version of the Cornell SVM-Light implementation [14] with a Gaussian radial basis function kernel and the term frequency of the keywords as features [15].

Anyway, we have discovered some limitations as soon as we apply SVM over real news sets. SVM has a strong dependence on the data used for training. While it works very well with texts dealing with highly general topics, it is not the case when we need to classify texts on very specific topics not included in the training stage, or when the main keywords change in the text over time. For example, when we talk about sports, every year sportsmen may belong to different teams in the same competition. If we want to use SVM we will need to change the training set at least every year. This means that the documentation department should manually label a lot of articles (for training) before the system could tag new articles properly. Therefore, we use SVM to categorize texts within high-level topics, but we also change the strategy later to obtain more detailed tags.

We have improved the SVM results by using techniques from Ontological Engineering [16]. We advocate the use of knowledge management tools (ontologies, semantic data models, and inference engines) due to the benefits that they can provide in this context. The first step is to design an ontology that describes the items we want to tag, but this is not an easy task in media business. The reason is that media cover many different themes, and therefore it is a disproportionate task to try to develop an ontology about all the publishable topics. We think that a better approximation is to design an ontology for every interesting subject

we want to tag, whenever SVM results are not able to get good results. In our experiments, we have selected the sports section, one of the most interesting for the audience, and at the same time one that has greater variability in terms of keywords from year to year. Inside this section, we have selected soccer articles as our target and we try to tag them automatically as best as the documentation department would do it. Therefore, our ontology includes teams, players, coaches, presidents, competitions, referees, and so on. Nevertheless, it is not only an ontology composed by named entities, as we have also introduced relationships and actions that join concepts providing semantic information, which is a substantial difference that brings advantages not contemplated by SVM. We have designed the ontology with the tool *Protégé* [18]. The ontology has been populated with current and rich data of the Spanish premier soccer league and stored in an OWL [19] file.

NASS tries to match each keyword with one of the words in the ontology. Before, we have prepared a table with the help of the documentation department and based on experimental and statistical analysis of the tags introduced manually. This table has two columns. In the first column we put ontology concepts. In the second column we put the probability of talking about a topic usually labeled with a term of the thesaurus when the system detects that concept in a text. Then, NASS submits SPARQL [20] queries against the ontology and it uses Jena [21] as framework and Pellet [22] as inference engine. As soon as it finds a keyword that matches a term of the ontology, it looks at its associated concept and then the system uses the previous table to retrieve the corresponding probability. At this point, it is important to mention that some keywords could be related with one or more thesaurus tags, and also a thesaurus tag could be related to one or more keywords. NASS increases the probability of tagging the article with a term each time it accesses a row of this table. A high number of accesses to the same thesaurus term guarantees that it can be used as a tag on the article. Through an extensive experimental evaluation we found useful to use 60% as a threshold to accept a term. Finally, NASS returns the thesaurus tags obtained by this method in order to label the text.

3 Experimental Evaluation

In this experimental evaluation, our goal is to evaluate the ability of the system to automatically annotate articles. All our experimentation has been performed with real sets of news belonging to a Spanish Media associated with the Vocento Group. We think the use of a standard corpus like the Reuters 25178, used in others approaches (for example, in [23] or [17]), is not so useful for us, as our objective is to solve a real problem with an own thesaurus and a particular way of labeling in a real documentation department. So, in our experiments we have used a corpus of 1755 articles, all of them tagged with thesaurus terms manually assigned by the documentation department (we will use them here in order to compare them with the automatic annotation performed by NASS). We consider that the input data set is a good representative of the types of articles managed.

We have applied lemmatization to the news corpus and then we have obtained keywords and named entities using the same value (0.5) for the parameters α and β (described in Section 2.3). Then, we have used SVM in order to find out the main topic of each article (for instance, to identify soccer articles). We trained NASS with thousands of sports articles from different years (already annotated). As one of the most important themes is the coverage of news related to soccer, which is the major sport in country, we refined the process applying NASS methods to decide which thesaurus terms were the best for labeling soccer articles. At this point, it is important to emphasize that the goal of our experimental evaluation was not only to detect which articles were talking about that topic, but also to properly label such articles by considering the suitable terms in the thesaurus.

We compared the tags obtained by our system with those that were manually assigned by the people working in the documentation department. The results are shown in Figure 2 and Figure 3. Specifically, in Figure 2 we consider a poorly populated ontology and in Figure 3 a properly populated one (with an increase of 25% of the terms). The results are presented using two confusion matrices that represent the items that are correctly tagged on the existing 1755 texts using the same ontology but with a different number of elements. We will obtain from these matrices two interesting facts: precision and recall. Generally in pattern recognition and information retrieval, precision is the fraction of retrieved instances that are relevant, while recall is the fraction of relevant instances that are retrieved. In our case we will assimilate "retrieved" to "properly tagged."

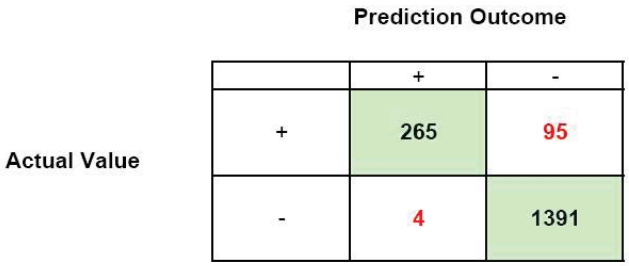


Fig. 2. Experimental results after using NASS with a poorly populated ontology

From the previous figures, several conclusions can be drawn. First of all, we found that the highest number of well-labeled articles occurs when we suitably populate the ontology, and if NASS fails it is due to a lack of semantic information (for example, when the name of a coach or a team president has not been introduced in the ontology). When the ontology has fewer elements, then the recall drops significantly (73% vs 95%) but the precision is the same (98%); indeed, it even improves due to the smaller chance of error because NASS gen-

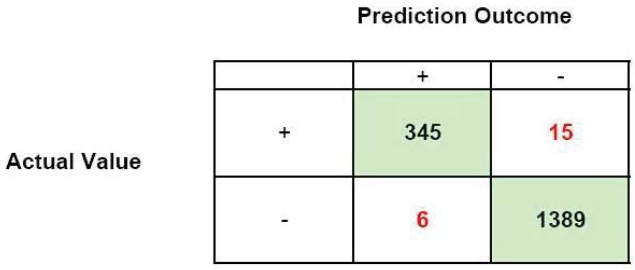


Fig. 3. Experimental results after using NASS with a properly populated ontology

erates a smaller number of labels. Anyway, the precision is good enough in both cases. Looking at the accuracy to put the tags, we could say that almost 99% of the labels generated by NASS were correct regardless of the number of elements that populate the ontology.

Summing up, the results obtained were really good. We obtained more than 95% of recall and precision. Furthermore, our system was able to detect additional labels that are relevant (even though they were not selected in the manual annotation) and avoid labels that were wrongly chosen by the documentation department.

4 Related Work

As commented along the paper, among other techniques, we have used a method commonly used for text categorization: the Support Vector Machine, or SVM [24]. This method has some appropriate features to face text classification problems, for example its capability to manage a huge number of attributes and its ability to discover which of them are important to predict the category of the text after a training stage. It is based on solid mathematical principles related to statistical learning theory and there are plenty of articles and books based on such mechanisms, not only for text classification but for any work related to cataloging all kinds of elements and entities. Our use of SVM is based on the references cited in Section 2.5.

Although OBIE is a relatively new field of study, most researchers believe that it could contribute very much to IE. Many researchers work with this kind of systems obtaining good results. So, in the last ten years, we can find an increasing number of articles facing IE problems by using OBIE systems. One of the earliest ones was *KIM* [25] (Knowledge Information Management). It was a generic solution and included an ontology, a knowledge base, a semantic annotation tool, and an indexing and retrieval server, as well as front-ends for interacting with the server. The developers of KIM used a corpus composed of

newspaper articles and the system extracted information using linguistic rules and gazetteer lists. Another early work came from Maedche et al. [26], with an OBIE system including an IE system able to adapt itself according to the ontology used. It used partial parse trees as the information extraction method. Another approximation is Embley's work [27], where heuristics and parsers are used to improve the information extractor using car advertisements as corpus. Regarding our problem, all these papers have been useful to a greater or lesser extent, but we have missed a real tagging over the data to compare the results with the desires of the future users of the system. McDowell and Caffarella [28] annotated web pages with their tool *OntoSyphon*, but with an ontology-guided process instead of going over the set of documents. That is an interesting point of view but without application for us due to the way the data come in newspapers: our system is clearly a document-pivoted categorization, instead of a category-pivoted categorization.

As an example of project combining SVM and ontologies we can reference [29], a recent work where SVM is used to categorize economic articles using multi-label categorization. The big difference is that they use ontologies to create labels prior to the categorization process, and then use different types of SVM only in that process, which does not let them obtain neither a high degree of accuracy nor a high number of categories, whereas our proposal avoids these problems. We would also like to mention the work performed by Wu et al. [30], which faced this kind of problems using a quite interesting unsupervised method based on a Naive-Bayes classifier and natural language processing techniques. They obtained good results and it has the advantages of being an unassisted process, but they do not meet the minimum requirements of precision and accuracy that were needed for our project.

5 Conclusions and Further Work

In this paper, we have presented an information extraction system that helps documentation departments of media companies in their daily annotation job when they use a thesaurus as the annotation tool. To evaluate the accuracy of the system, we have performed experiments on real news previously tagged manually by the staff of the documentation department. This work has contributed to make a good adaptation of the general methods of SVM and OBIE systems over a real and changing set of news instead of the typical and less realistic standard news corpus. Our experimental results show that we are able to get a reasonable number of correct tags using IE methods like SVM, but the accuracy improves with the combined use of NLP and semantic tools when the training set of the SVM must be updated frequently (e.g., in the context of sports, where the specific data change every season). Instead of having to label news each year to create a reliable set for training, we propose that documentalists fill the instances of classes in a predefined ontology. Then, our system has enough information to label the news automatically by using semantic tools. We found this simpler and more intuitive for end users, and it helps to get better results.

Besides, the accuracy of the automatic assignment of tags with our system is very good, obtaining 99% of correct labels. In fact, the current version of NASS is already being successfully used in several companies. This shows the interest of our approach.

As an evaluation scenario, we have considered so far news related to sports. In the future, we will continue applying NASS to other news sections to verify its operation. We also plan to introduce new and more powerful methods to enrich the system providing it with greater speed, wider scope and better accuracy. Priority for us is also reducing the need for manual intervention during the process, an aspect in which we are already working. One of the most important challenges in this area is to develop a real, intelligent, useful and efficient semantic tool in the always-changing environment of the media, ready to help categorization tasks in archives and useful to distribute news in all the Internet formats: web, phones, tablets, and so.

References

1. A. Gilchrist. 2003. Thesauri, taxonomies and ontologies: an etymological note. Journal of Documentation vol. 59(1): pp. 7-18.
2. A. L. Garrido, O. Gomez, S. Ilarri and E. Mena. 2011. NASS: news annotation semantic system. Proceedings of ICTAI 11, International Conference on Tools with Artificial Intelligence: pp. 904-905. IEEE.
3. A. F. Smeaton. 1997. Using NLP or NLP resources for information retrieval tasks. Natural Language Information Retrieval. Kluwer Academic Publishers.
4. T. Joachims. 1998. Text categorization with support vector machines: learning with many Relevant Features. Proceedings of ECML 98, European Conference on Machine Learning: pp. 137-142. Springer.
5. D.C. Wimalasuriya and D. Dou. 2010. Ontology-Based Information Extraction: an introduction and a survey of current approaches. Journal of Information Science vol. 36(3): pp. 306-323. Sage Publications.
6. A. L. Garrido, O. Gomez, S. Ilarri and E. Mena. 2012. An Experience Developing a Semantic Annotation System in a Media Group. Proceedings of NLDB 12, International Conference on Applications of Natural Language Processing to Information Systems: pp. 333-338. Springer.
7. S. Russell and P. Norvig. 2003. Artificial Intelligence: a modern approach. Prentice-Hall.
8. T.R. Gruber. 1993. A translation approach to portable ontology specifications, Knowledge Acquisition vol. 5(2): pp. 199-220. Academic Press Ltd.
9. G.A. Miller. 1995. WordNet: a lexical database for english. Communications of ACM vol. 38(11): pp. 39-41. ACM.
10. X. Carreras, I. Chao, L. Padró, and M. Padró. 2004. Freeling: an open-source suite of language analyzers. Proceedings of the 4th International Conference on Language Resources and Evaluation: pp. 239-242. European Language Resources Association.
11. P.A. Chirita, S. Costache, W. Nejdl, and S. Handschuh. 2007. P-tag: large scale automatic generation of personalized annotation tags for the web. Proceedings of WWW 07, International Conference on World Wide Web: pp. 845-854. ACM.
12. G. Salton and C. Buckley. 1988. Term weighting approaches in automatic text retrieval. Information Processing and Management, vol. 24(5): pp. 513-523. Pergamon Press, Inc.

13. S. Sekine and E. Ranchhod. 2009. Named entities: recognition, classification and use. John Benjamins.
14. T. Joachims. 2004. SVM-Light version 6.01. Ithaca, NY: Department of Computer Science, Cornell University.
15. E. Leopold and J. Kindermann. 2002. Text categorization with support vector machines. How to Represent Texts in Input Space? Machine Learning vol. 46: pp. 423-444. Kluwer Academic Publishers.
16. M. Fernandez-Lopez and O. Corcho. 2004. Ontological engineering. Springer.
17. G. Schohn and D. Cohn. 2000. Less is more: Active learning with support vector machines. Proceedings of the 17th International Conference on Machine Learning: pp. 839-846. Morgan Kaufmann.
18. N.F. Noy, M. Sintek, S. Decker, M. Crubézy, R.W. Fergerson and M.A. Musen. 2001. Creating semantic web contents with Protégé-2000. IEEE Intelligent Systems vol. 16(2): pp. 60-71. IEEE.
19. D.L. McGuinness, F. van Harmelen. 2004. OWL Web Ontology Language overview. W3C Recommendation. Available at: `http://www.w3.org/TR/owl-features/`
20. E. Prudhommeaux, A. Seaborne. 2006. SPARQL Query language for RDF. W3C Working Draft. Available at: `http://www.w3.org/TR/2006/WD-rdf-sparql-query-20061004/`
21. J.J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. 2004. Jena: Implementing the semantic web recommendations. Proceedings of WWW 04, International Conference on World Wide Web: pp. 74-83. ACM.
22. E. Sirin, B. Parsia, B. Cuenca, A. Grau, Y. Kalyanpur. 2007. Pellet: a practical OWL-DL reasoner. Journal of Web Semantics vol. 5(2): pp. 51-53.
23. A. Moschitti and R. Basili. 2004. Complex linguistic features for text classification: a comprehensive study. Proceedings of ECIR 2004, European Conference on Information Retrieval: pp. 181-196. Springer.
24. C. Cortes, V. N. Vapnik. Support-vector networks. Machine Learning vol. 20(3):273-297. Kluwer Academic Publishers.
25. B. Popov, A. Kiryakov, A. Kirilov, D. Manov, D. Ognyanoff, and M. Goranov. 2003. KIM - semantic annotation platform. Natural Language Engineering vol. 10(3-4): pp. 375-392. Cambridge University Press.
26. A. Maedche, G. Neumann, S. Staab. 2003. Bootstrapping an ontology based information extraction system. Springer.
27. D.W. Embley. 2004. Toward semantic understanding: an approach based on information extraction ontologies. Proceedings of the 15th Australasian Database Conference: pp. 3-12. Australian Computer Society, Inc.
28. L. K. McDowell and M. Cafarella. 2006. Ontology-driven information extraction with OntoSyphon. Proceedings of the 10th International Semantic Web Conference: pp. 428-444. Springer.
29. S. Vogrincic and Z. Bosnic. 2011. Ontology-based multi-label classification of economic articles. Computer Science and Information Systems, vol. 8(1): pp. 101-119. ComSIS Consortium.
30. X. Wu, F. Xie, G. Wu, W. Ding. 2011. Personalized news filtering and summarization on the web. Proceedings of ICTAI 11, International Conference on Tools with Artificial Intelligence: pp. 414-421. IEEE.

Sesión Temática 2

Ingenería Web, Interfaces de Usuario, Sistemas Colaborativos, Computación Ubicua

Coordinadores: *Dr. Pascual González y Dr. Juan Carlos Preciado*

Sesión Temática 2: Ing. Web, Interf. Usuario, Sist. Colaborativos, Computación Ubicua
Coordinadores: Dr. Pascual González y Dr. Juan Carlos Preciado

Miguel Sánchez Román, et al. *Políticas de seguridad en sistemas workflow colaborativos*. (Emergente)

Joaquina Martin-Albo and Coral Calero. *Redes Sociales: Estrategia de Marketing para la pequeña empresa*. (Emergente)

Jesus M. Hermida, et al. *Sm4RIA Extension for OIDE: Desarrollo de Rich Internet Applications en la Web Semántica*. (Herramienta)

Victor M. R. Penichet, Maria-Dolores Lozano and Jose A. Gallud, Ricardo Tesoriero. *TOUCHE CASE Tool: A Task-Oriented and User-Centered Case Tool to Develop Groupware Applications*. (Herramienta)

Miguel A. Teruel, et al. *CSRML Tool: una Herramienta para el Modelado de Requisitos de Sistemas Colaborativos*. (Regular)

Natalia Padilla-Zea, et al. *Una arquitectura para el desarrollo de videojuegos educativos con actividades colaborativas*. (Regular)

Francy D. Rodríguez and Silvia T. Acuña. *Implementación de una Solución Reutilizable para una Funcionalidad de Usabilidad*. (Regular)

Juan Antonio Pereira, et al. *An experience migrating a Cairngorm based Rich Internet Application from Flex to HTML5*. (Regular)

Iñaki Fernández De Viana Y González, et al. *Verificación de la información extraída por wrappers web usando algoritmos basados en colonias de hormigas*. (Regular)

Francisco Montero, et al. *Computer-Aided Relearning Activity Patterns for People with Acquired Brain Injury*. (Relevante)

Alejandro Catala, et al. *Exploring Tabletops as an Effective Tool to Foster Creativity Traits*. (Relevante)

Juan Carlos Preciado. *Tutorial: Desarrollo Dirigido por Modelos en Ingeniería Web con Webratio y RUX-Tool*. (Tutorial)

A. Ruíz, L. Iribarne (Eds.): Actas de las "*XVII Jornadas de Ingeniería del Software y Bases de Datos (JISBD'2012)*", Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

Políticas de Seguridad en Sistemas Workflow Colaborativos

M. Sánchez Román [1], B. Jiménez Valverde [1], F.L. Gutiérrez Vela [1], P. Paderewski [1]

[1] Departamento de Lenguajes y Sistemas Informáticos ,Universidad de Granada.
{miguesr,beajv,fgutierr,patricia}@ugr.es

Abstract. En los sistemas workflow colaborativos, además de los requisitos de seguridad tradicionales como la disponibilidad, la integridad y la confidencialidad, nos encontramos con una serie de requisitos de seguridad específicos que deben considerarse. La inclusión de estos requisitos de seguridad es un proceso costoso que puede reducirse si se tienen en cuentan la definición y la coordinación de políticas de seguridad específicas para estos sistemas. En este trabajo presentamos un análisis inicial de los requisitos que consideramos son necesarios para la definición de políticas de seguridad en workflows colaborativos dentro del marco de una arquitectura basada en servicios web con objeto de facilitar la especificación, gestión, y verificación de dichas políticas.

Keywords: Sistemas colaborativos, Workflow, RBAC , Políticas de seguridad

1 Introducción

El auge del uso de recursos y servicios en la "nube", las organizaciones virtuales y los servicios de Internet (redes sociales, conferencias online, e-commerce, e-learning) demuestra que las aplicaciones de negocio no pueden seguir un modelo de vida tradicional, que es el de desarrollar y ofrecer servicios IT dentro del marco empresarial u organizacional. Las organizaciones deben redefinir sus procesos de negocio, tanto desde el punto de vista estratégico como técnico para que sus servicios sean proporcionados en un entorno web y en muchos casos bajo una filosofía de trabajo colaborativo y con un fuerte uso de redes sociales. La definición y gestión de requisitos de seguridad, en la redefinición de estos procesos de negocio en entornos web, requiere una serie de esfuerzos para garantizar la confidencialidad, integridad y disponibilidad del servicio que resultan, en muchos casos, incluso difíciles de verificar. Esta gestión se vuelve aún más compleja cuando se trata del caso particular de los procesos de negocio colaborativos donde surgen una serie de requisitos característicos.

Modelos de seguridad que tradicionalmente se usan en sistemas de información organizacionales como DAC (*Discretionary Access Control*), MAC (*Mandatory Access Control*) y RBAC (*Role-Based Access Control*) no son capaces de cubrir los requisitos de seguridad de los sistemas workflow colaborativos. En los modelos DAC los permisos de acceso son definidos a discreción del usuario. En MAC, las decisiones de seguridad no recaen en el usuario, y es el sistema el que fuerza el cumplimiento de las

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 111-106, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

políticas por encima de las decisiones de los usuarios. En el modelo RBAC las decisiones de acceso están basadas sobre los roles que los usuarios adquieren como parte de la organización, permitiendo definir políticas de seguridad mas complejas que los modelos DAC y MAC, no obstante, como se detallará más adelante en este trabajo, tampoco permite definir algunas características de seguridad necesarias en workflows colaborativos.

En la Sección 2 examinamos el concepto de workflow centrando nuestra atención en los de carácter colaborativo y sus necesidades de coordinación. En la Sección 3 presentamos las políticas de seguridad en los sistemas de información y en la Sección 4, examinamos las características y requisitos adicionales que son necesarios para la definición de políticas de seguridad en sistemas workflow colaborativos. En la Sección 5 presentamos una aproximación inicial respecto a cómo integrar las políticas de seguridad dentro de una arquitectura basada en servicios web con objeto de facilitar su gestión, especificación y verificación. Finalmente presentamos las conclusiones.

2 Necesidades de Coordinación en Workflows Colaborativos

La WfMC (Workflow Management Coalition) [1] define un proceso de negocio como: “Un conjunto de uno o más procedimientos o actividades directamente ligadas, que colectivamente realizan un objetivo del negocio, normalmente dentro del contexto de una estructura organizacional que define roles funcionales y relaciones entre los mismos." Igualmente la WfMC define un workflow como: "La automatización de un proceso de negocio, total o parcial, en la cual documentos, información o tareas son pasadas de un participante a otro a los efectos de su procesamiento, de acuerdo a un conjunto de reglas establecidas”.

Los workflows se clasifican en [2]: workflows de producción, workflows de colaboración y workflows de administración. En este trabajo nos centramos en los workflows de colaboración, éstos estructuran procesos de negocio donde participan personas que realizan distintas actividades con el objetivo de lograr una meta común.

A la hora de definir un workflow colaborativo, la interacción grupal se convierte en un elemento clave [3]. Para alcanzar una coherencia de la interacción grupal se debe prestar mucha atención a la coordinación. Es decir, la efectividad de la colaboración tendrá lugar siempre que las actividades del grupo estén bien coordinadas.

En un workflow colaborativo los usuarios actúan bajo un rol determinado. Cada rol tiene una responsabilidad en el proceso de negocio. Coordinando las habilidades de cada rol se consigue la finalidad del flujo de trabajo.

Las restricciones o permisos de acceso de cada rol en el flujo de la coordinación deben definirse mediante la combinación de la definición de actividades o procedimientos y las definiciones de roles. Esta combinación requiere ser definida con un nivel de grano fino como consecuencia directa del dinamismo propio de los sistemas colaborativos. Un escenario donde sería preciso establecer un control de grano fino es el caso de un entorno hospitalario cuando se crea un grupo de trabajadores sanitarios para dar asistencia médica a un paciente en concreto. En este caso sólo ciertos miembros de este grupo podrán tener acceso al expediente del paciente, como por ejemplo

un rol Medico o un rol Enfermero. Sin embargo, otros miembros de ese grupo, como por ejemplo el rol Celador, no tienen por qué tener acceso al expediente del paciente.

3 Políticas de Seguridad

La definición de las políticas de seguridad en el desarrollo de un sistema es una tarea compleja, ya que en su especificación es necesario considerar diversos aspectos como son: los objetivos del negocio, la dinámica de cooperación, la regulación de las responsabilidades, los conflictos de intereses, las condiciones impuestas por las instituciones gubernamentales, así como otros factores del entorno de la organización. Las políticas de seguridad se catalogan en Discrecionales o No Discrecionales.

Control de Acceso Discrecional (DAC – Discretionary Access Control). En las políticas DAC los permisos para el control de acceso se definen a discreción por los propietarios de los objetos y cualquier usuario que esté autorizado a controlar el acceso del objeto [4]. Por ejemplo, estas políticas se usan de forma genérica para limitar a un usuario el acceso a un archivo [5]. El propietario del archivo es quien controla el acceso de otros usuarios a éste.

Control de acceso no discrecional (NDAC – Non-Discretionary Access Control). Dentro de esta categoría nos encontramos con los siguientes tipos de políticas:

Las *Políticas de Separación de deberes (SOD – Separation Of Duty)***.** Se utilizan en organizaciones para asegurar la integridad de los procesos de negocio, guiando y evadiendo de forma apropiada las situaciones de conflictos de intereses. Existen diferentes formas de políticas de tipo SOD [6][7][8][9][10], las más relevantes son:

SOD estáticas basadas en roles: Dos roles nunca pueden ser asignados a la misma persona. Por ejemplo, un usuario no puede ser a la vez el contable y el gerente.

SOD dinámicas basadas en roles: Nos permiten definir que dos roles con conflictos de intereses en un proceso de negocio no pueden ser asignados o activados por el mismo usuario en la misma instancia del proceso pero sí en distintas instancias.

SOD basadas en la identidad: Permiten definir reglas para evitar que a determinados usuarios, se le asigne un rol concreto. Por ejemplo, que a un empleado de dudosa honorabilidad no se le asigne el rol administrador.

SOD basadas en objetos: Permiten especificar que un usuario no pueda realizar múltiples operaciones sobre un mismo objeto participando en dos roles distintos. En la realización de un pedido de compra, éste no puede ser preparado y firmado por el mismo responsable.

SOD operacional: Define que un solo usuario participando en un rol no puede realizar todas las actividades relacionadas con un proceso de negocio. Estas políticas se centran en las actividades dentro del ciclo de vida de un proceso de negocio.

SOD basada en la historia: Estas políticas están basadas en información histórica, por ejemplo, el orden predefinido sobre las acciones realizadas por los roles, el número de veces que un rol ha accedido a un mismo objeto, etc.

Las *Políticas de Mínimos Privilegios.* (*LP – Least Privilege*) establecen que un usuario no puede obtener más privilegios que los estrictamente necesarios para la realización de sus actividades [11].

Las *Políticas de control de acceso obligatorio (MAC – Mandatory Access Control) s*e caracterizan porque la decisión de definición de permisos de control de acceso es realizada por una autoridad central, no por el usuario propietario de un objeto, y asegurando que el propietario del objeto no puede cambiar los permisos de acceso [12].

En las *Políticas de control de acceso basadas en roles (RBAC – Role-Based Access Control)* las decisiones de acceso están basadas en los roles [13] que los usuarios adquieren como parte de la organización. Cada usuario es miembro de un conjunto de roles conforme a sus capacidades, competencias y responsabilidades dentro de la organización, y los permisos de acceso se agrupan por rol; el acceso a los recursos está restringido al usuario conforme al rol que éste tiene asociado.

En las *Políticas basadas en restricciones de tiempo (Temporal Constraints)* se usan reglas formales que involucran restricciones basadas en el tiempo sobre los permisos de acceso a los recursos. Estas políticas se utilizan en un gran número de aplicaciones donde se definen restricciones temporales en el uso de los recursos, en la realización de actividades o en la asignación de usuarios a roles.

Las *Políticas Chinese Wall* [14] fueron identificadas por Brewer and Nash para solucionar los conflictos de intereses relacionados con actividades de consulta dentro de organizaciones bancarias y de otras disciplinas financieras. Por ejemplo, un consultor para realizar un servicio a sus clientes tiene que tener acceso a la información protegida de su cliente. Esto implica que el consultor obtiene conocimiento de información confidencial del cliente y que éste puede ser usado fuera de la compañía, para socavar la ventaja competitiva de una institución, o usarlo con propósitos personales.

4 Requisitos para la Definición de Políticas de Seguridad en Workflows Colaborativos

En el apartado anterior hemos realizado una presentación de las políticas más comunes para el control de acceso en los sistemas de información de las organizaciones. Sin embargo, estas políticas no permiten expresar en su totalidad las características y requisitos propios de los workflow colaborativos así como su inherente dinamismo. Algunas características y requisitos, que son necesarios para la definición de políticas de control de acceso en sistemas workflow colaborativos, son:

- Necesidad de **manejar información sensible al contexto del entorno colaborativo**. Esta información de contexto puede contemplar información tan variada como la localización física o el estado de coordinación de los usuarios. Por ejemplo, en un sistema de votación se puede establecer la política de que los usuarios con rol "Votante" no podrán realizar su derecho de votación hasta que todos los miembros participantes en dicha votación estén presentes.
- Necesidad de considerar que en los sistemas workflow colaborativos **los permisos y privilegios de los usuarios pueden variar conforme progresan las actividades** que componen el flujo colaborativo. Estos cambios pueden deberse a acciones propias de los usuarios o simplemente al estado actual del proceso. Por ejemplo, en un proceso de gestión de gastos en un almacén solo después del envío del informe de inventario, por parte del rol "Responsable de Inventario", el rol "Contable" podrá ver dicha información, mientras tanto no tendrá acceso a la información de inventario.

- Otro requisito es que **solo el propietario de un rol o grupo de roles conozca las identidades de los miembros que actúan bajo ese rol**. Por ejemplo, en un proceso de revisión de artículos de una conferencia, un miembro del rol "Comité del Programa" es propietario de un conjunto de roles "Revisor" encargados de revisar los artículos enviados que él gestiona. Sin embargo, ningún otro usuario que tenga el rol "Comité de Programa" tiene permitido ver las identidades de los miembros del rol "Revisor" de los que no es propietario. Es decir, es necesario poder especificar políticas control de sobre usuarios individuales en ciertos roles.
- Es necesario que las políticas consideren la **información de las interacciones entre los diferentes usuarios y los procesos del sistema**. Por ejemplo, en un proceso de realización de un trabajo en grupo, se establece que los estudiantes solo pueden ver las especificaciones del trabajo una vez que el rol "Profesor" haya entregado a todos los roles "Alumno" el enunciado del trabajo.

5 Integración de Políticas de Seguridad dentro de una Arquitectura Web.

En trabajos previos hemos presentado la descripción de un modelo abstracto que permite describir los elementos que intervienen en un sistema colaborativo así como la definición de un conjunto de servicios web con objeto de gestionar la coordinación en base a esos modelos de manera que sea posible y efectiva la colaboración [15]. Una vez analizadas las necesidades concretas de los sistemas workflow colaborativos respecto a aspectos de seguridad, es necesario representar estas necesidades usando un modelo que nos permita definir fácilmente las políticas de seguridad. Para la representación de los aspectos de seguridad partimos del modelo RBAC añadiendo los elementos necesarios para describir las políticas de seguridad descritas en este trabajo (separation of duties, políticas de acceso basadas en restricciones de tiempo, políticas sensibles al contexto, …).

Para la gestión, definición y verificación de las políticas de seguridad, en nuestra propuesta de arquitectura basada en servicios web para el soporte de sistemas colaborativos, será necesario incluir un conjunto de servicios que lleven a cabo estas tareas, tales como: un **Servicio Web de autorización**, que se va a encargar de almacenar y gestionar la información relativa a las políticas de autorización implementadas en el sistema; un **Servicio Web de validación de políticas**, que será el encargado de validar la integridad de las políticas evitando incongruencias entre las políticas definidas; Por último, un **Servicio Web de definición de políticas,** que nos facilitará las herramientas necesarias para definir y activar las políticas a aplicar en la organización.

6 Conclusiones

En este trabajo hemos realizado un análisis de las políticas de seguridad más comunes en los sistemas de información de las organizaciones. También hemos analizado los requisitos de seguridad que deben ser considerados en los sistemas workflow colaborativos. Como conclusión de ambos análisis consideramos necesaria la inclu-

sión de nuevos elementos que nos permitan expresar en su totalidad los requisitos de seguridad de los workflow colaborativos. Para la representación de los aspectos de seguridad partimos del modelo RBAC. Integramos las políticas de seguridad dentro del marco de una arquitectura basada en servicios web para facilitar su gestión, definición y verificación.

Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Ciencia e Innovación de España como parte del proyecto VIDECO (TIN2011-26928).

Referencias

1. "Workflow Management Coalition Standard – Terminology and Glossary", Workflow Management Coalition (WfMC). (WFMC-TC-1011, Feb-1999, 3.0)
2. Edward A. Stohr and J. Leon Zhao.: "Workflow Automation: Overview and Research Issues", Information Systems Frontiers: Special Issue on Workflow Automation and Business Process Integration, Vol. 3, Issue 3, pp. 281-96 (2001)
3. Ellis, C.A., Gibbs, S.J., Rein, G.L.: "Groupware: Some Issues and Experiences". Communications of the ACM, Vol. 34, No. 1 , pp. 38-58 (1991)
4. National Computer Security Center (NCSC), "A Guide to Understanding Discretionary Access Control in Trusted System," Report NSCD-TG-003 Version1 (1987)
5. NIST Special Publication 800-7, National Institute of Standards and Technology. (1994)
6. Gligor, V. Gavrila, S., Ferraiolo, D., "On the formal definition of separation-of-duty policies and their composition". Proceedings IEEE Symposium on Security and Privacy. IEEE Computer Society Press, pp. 172-183 (1998)
7. Nyanchama, M. Osborn. S, "The Role Graph Model and Conflict of Interest". ACM Transaction on Information System Security 2 (1), pp. 3-33 (1999)
8. Sandhu, R. S., "Transaction control expressions for separation of duties". In: Fourth Annual Computer Security Application Conference. pp. 282-286 (1988)
9. Simon R.T., and Zurko M. E., "Separation of Duty in Role-Based Environments," Proc. of the Computer Security Foundations Workshop X, Rockport, Massachusetts (1997)
10. Tidswell, J.E., Jaeger, T. "Integrated constraints and inheritance in DTAC" In: ACM Workshop on Roles-based Access Control. pp. 93-102 (2000)
11. Saltzer J., Schroeder M. "The Protection of Information in Computer Systems" Communications of the ACM 17, 7 (1974).
12. Bell DE, LaPadula LJ, "Secure Computer Systems: Mathematical Foundations and Models." Mitre Report M74–244, Mitre Corporation, Bedford, Massachussets (1974)
13. Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C, "Role-based access control models". *Computer* 29, 2, pp. 38-47 (1996)
14. Brewer D., and Nash M., "The Chinese Wall Security Policy," Proc IEEE Symp Security & Privacy, IEEE Comp Soc Press, pages pp. 206-214 (1989)
15. Jimenez B., Sánchez M., Gutierrez F.L., Paderewski P.: "A Service Oriented Architecture for Coordination in Collaborative Environments", International Journal of Information Technologies and the systems Approach (IJITSA), vol. 4, Issue 1, pp. 79-92 (2011)

Redes Sociales: Estrategia de Marketing para la pequeña empresa

Joaquina Martín-Albo, Coral Calero

Instituto de Tecnologías y Sistemas de Información (ITSI)
Universidad de Castilla-La Mancha
jmartinalbo@hotmail.com; Coral.Calero@uclm.es

Resumen. En la última década las redes sociales se han convertido en el fenómeno de moda en Internet, abarcando prácticamente todos los ámbitos de la actividad humana, y permitiendo alinear voces, conciencias y acciones de manera simple, lo cual se ve reflejado en el mundo empresarial de forma directa, ya que los clientes pueden ejercer fuerzas a favor o en contra de las marcas y compañías a una velocidad vertiginosa. En este nuevo escenario toda empresa, y en particular la pequeña empresa, debe hacer notar su presencia en las mismas, con el fin de que el cliente se sienta partícipe de nuestro plan de negocio. Con este trabajo, nuestra intención, es mostrar la evolución de las redes sociales en España en los últimos años, para proponer una Estrategia de Marketing que partiendo de las preferencias y comentarios de los españoles en estos entornos, permita integrar a un pequeño negocio en la plataforma social adecuada, a un bajo coste, pero con una óptima capacidad de atracción del consumidor, ya que hasta el momento no existen líneas directrices de cómo adoptar el uso de las Redes Sociales en Pymes.

Palabras Clave. Redes Sociales, evolución, marketing digital

1. Introducción

El establecer y mantener relaciones sociales con otros miembros de nuestra especie es algo innato en el ser humano. Con la explosión de la Web Social [7] se ha ido incorporando esta filosofía en numerosos sitios web, de manera que cada vez es más frecuente encontrar servicios que permitan mantener una red de contactos con los que comunicarse y compartir recursos (comentarios, imágenes, vídeos,…). De este modo las redes sociales emergen constituyendo un nuevo espacio de comunicación humana tan singular que transforma las relaciones empresa-cliente y marca-consumidor, modificando la praxis del Marketing [4], por lo que el empresario se encuentra con la ardua tarea de identificar las oportunidades que le brindan las redes sociales para mejorar la relación con sus exigentes clientes.

España, y más en la dificultad económica en que nos encontramos, también ha de explotar el momento de auge que viven en nuestro país estos sitios web, para potenciar y lanzar sus estrategias de negocio.

Aunque a un alto nivel empresarial se cuenta con la figura del Community Manager, encargado de gestionar la información procedente de redes sociales, en este trabajo nos centramos en la integración de estas plataformas sociales en pequeñas empresas a

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 117-122, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

bajo coste, pero con la misma capacidad de atracción del consumidor, dada la ausencia de estudios enfocados al uso de las Redes Sociales en Pymes.
En el punto 2 se dará una clasificación de redes sociales, para pasar a reflejar la evolución de estas plataformas en España a lo largo de los últimos años, con el fin de facilitar a la pequeña empresa la predicción del comportamiento de los españoles a la hora de utilizar estos medios. Por último se presenta una posible estrategia de marketing a seguir por un negocio interesado en integrar las redes sociales como parte de sus herramientas de viabilidad comercial.

2. Clasificación de Redes Sociales.

Las redes sociales han sufrido numerosos cambios a lo largo de su corta historia, lo que refleja el dinamismo de este sector. En [6], se presenta una clasificación ortogonal en base a dos dimensiones, que quedan fusionadas en la matriz de clasificación de las mismas (ver Figura 1):

- **Público Objetivo:**

-*Redes Horizontales:* Dirigidas a todo tipo de usuario, que no busca un tema concreto.
-*Redes Verticales:* Dirigidas a un público reunido en torno a un interés específico.

- **Propósito Final de la red:**

-*Redes Sociales:* Generación de una gran masa de usuarios que formen parte de la red.
-*Redes Profesionales:* Agrupar usuarios que aporten información valiosa a la red a la hora de satisfacer nuestros objetivos profesionales.

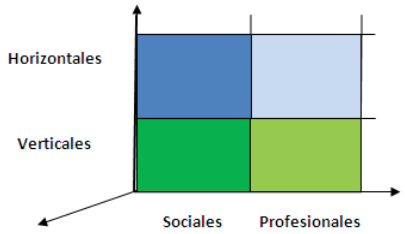


Figura 1. Matriz de Clasificación de Redes Sociales

2.1. Redes Sociales más utilizadas en España.

Presentamos en este punto la evolución de las redes sociales en nuestro país a lo largo de los cuatro últimos años, con el fin de darle a conocer a la pequeña empresa hacia donde se dirigen los intereses de los españoles y poder atraer y captar futuros clientes a sus negocios [9]. Desde el año 2008 contamos con estudios realizados por IAB Spain (Interactive Advertising Bureau), Elogia[1] [1], [2], [3] y por [5] sobre las 10 redes sociales más populares en nuestro país, mostradas en las figuras 2 a 5 en orden decreciente de número de usuarios registrados.

[1] Empresa de marketing digital que innova con las últimas tendencias online para dar respuesta a nuestros objetivos.

Así, como podemos observar en las matrices Facebook, Tuenti y Twitter, ocupan, en España, un lugar destacado entre las redes sociales más populares (en el año 2011 entre las tres alcanzaron un total de 20 millones de usuarios tan solo en nuestro país [15]), quedando patente nuestro interés por las redes horizontales sociales y de redes verticales sociales como Badoo, como la manera de formalizar estos contactos. Además, se ha detectado que existe poca variación en las redes utilizadas a lo largo de estos años, lo cual indica una cierta "fidelidad" de los usuarios españoles a las redes sociales escogidas. Una vez consolidades las redes como parte activa de nuestra vida personal, el intento de extrapolar esta funcionalidad a nuestro ámbito profesional, hace que comienze a haber movimiento en Redes Horizontales Profesionales como LinkedIn que facilitan la vida laboral y donde la empresa encuentra un buen marco de desarrollo de sus espacios estratégicos.

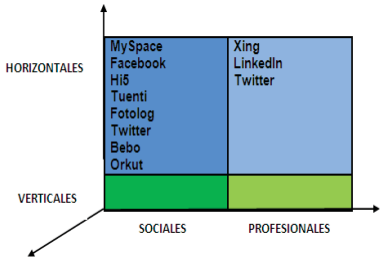


Figura 2. Redes Sociales más utilizadas en España en 2008.

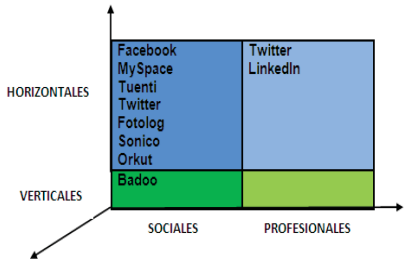


Figura 3. Redes Sociales más utilizadas en España en 2009.

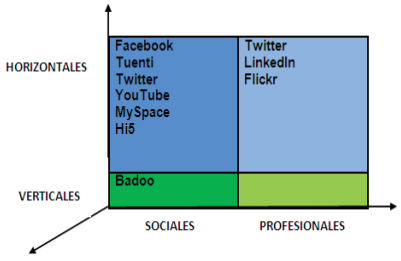


Figura 4. Redes Sociales más utilizadas en España en 2010.

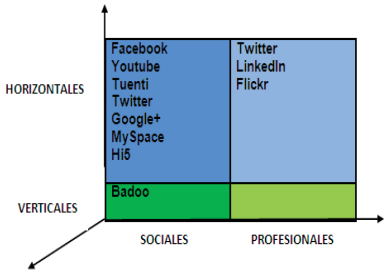


Figura 5. Redes Sociales más utilizadas en España en 2011.

Parece, por otro lado, que aún no estamos sensibilizados al potencial que nos ofrecen las redes Verticales Profesionales, donde no destaca ninguna red por la cantidad de usuarios registrados. Esto puede ser debido a la diferencia de usuarios potenciales, ya que de ningún modo son comparables los usuarios de una red general con los que pueda tener una red dirigida a un público específico. Por tanto, queremos aquí indicar que ya contamos con redes verticales profesionales en español (como Pleiteando) que comienzan a emerger.

Queda reflejado que las redes sociales en España se usan, en sus cuatro dimensiones, si bien quedan aún celdas de la matriz que pueden ser más "explotadas".

3. Redes Sociales: Factor de Competitividad entre Mercados.

Como ha quedado mostrado en el punto anterior en nuestro país las redes sociales están siendo cada vez más incorporadas como un nuevo modo de comunicación, tanto a nivel personal como profesional. Por ello muchas negocios se han incorporado al mundo de las redes sociales con la idea de convertirse en otro nodo fundamental de la red, pero sin saber muy bien cómo hacerlo.

Aunque es evidente que el hecho de contar con una red social no garantiza el éxito o recuperación de una empresa (como ocurriera en su día con las páginas web) si pueden ayudar como una herramienta de marketing de gran potencial, que les permitirá a un bajo coste reforzar su posición en el mercado, llegando a clientes objetivo con los que de otro modo les hubiera resultado complicado o demasiado caro contactar. Señalar aquí, el hecho de que esta Internet Social es de conversaciones y no monetizable, pero supone una fuente de captación de nuevos clientes hacia nuestros negocios, dónde si se hace tangible esta relación [8].

El estudio realizado sobre redes sociales en [12] nos ha llevado a marcar las siguientes líneas de marketing a la hora de utilizar estos sitios web en beneficio de nuestro negocio:

-Definición del objetivo/os buscados a la hora de adentrarse en el mundo de las redes sociales: captación, fidelización de clientes, reconocimiento de marca, canal de sugerencias…

-Analizar nuestro entorno: qué redes sociales son las más utilizadas del momento, en cuáles se concentra más cantidad de nuestro público objetivo…Este estudio queda reflejado en nuestra tabla de utilización de redes sociales, donde se detecta la importancia de las redes Horizontales tanto Sociales como Profesionales.

-En función de las conclusiones sacadas hasta este momento, estudiaremos que red/es sociales se adaptan más al negocio, ya que cada una ofrece distintas acciones de marketing a explotar, lo cual requiere un conocimiento exhaustivo de la filosofía seguida por la misma. Por ejemplo, si optamos por *redes horizontales* como Facebook, debemos conocer que esta red cuenta con páginas de empresa (con las que comunicarnos con nuestros clientes y ofrecer información de interés). Si optamos por el empleo de *redes verticales*, hemos de ser más cuidadosos en la información aportada, dado que nos dirigimos a un público más especializado.

-Agudizar el ingenio para ganar seguidores lo que aumentará nuestro prestigio corporativo y éste nos ayudará a conseguir clientes potenciales. Se trata de conseguir que sean ellos los que hablen de nosotros en lugar de nosotros mismos.

-Detectar necesidades. Algunas redes sociales como Twitter cuentan con un buscador interno que permite monitorizar en tiempo real Tweets con palabra clave. Por tanto, se

observa la necesidad de implementar una herramienta que permita extraer preferencias de los comentarios aportados por los usuarios en las redes y así captar futuros clientes que esten demandando nuestro producto o servicio. De este modo queda reflejado como la red social es un mecanismo de captación de posibles clientes hacia mi negocio, pero no son el sitio donde se realiza el mismo.

-La difícil pero recompensada tarea de conocer a nuestros usuarios, nos permitirá la creación de diferentes tipos de perfiles, y por tanto el poder dirigir nuestros anuncios a cada usuario de un modo personalizado.

-Por último y no menos importante contar con una buena dosis de compromiso y paciencia. No por estar en redes sociales nuestro negocio subirá de forma exponencial.

El medir el llamado ROI[2] en estos medios sociales no es fácil sobre todo teniendo en cuenta que muchas veces se orienta hacia aspectos intangibles, como la imagen de la marca. Muchos de los sitios ofrecen herramientas que permiten medirlo, como el servicio Twitter analytics o las páginas de negocio de Facebook. En [10] se identifican cinco métricas que sirven para saber si una empresa está recibiendo los resultados esperados en esta inmersión.

Así, la estrategia planteada (ver Figura 6), no requiere una gran inversión económica, pero si una importante inversión de nuestro tiempo de análisis y de constancia, hasta conseguir ver reflejado este planteamiento en la economía de nuestra empresa.

4. Conclusiones y Trabajo Futuro.

La integración de las redes sociales en las empresas, y en concreto en las pequeñas empresas, previo diseño de una adecuada estrategia de Marketing, las puede convertir en potentes herramientas capaces de lanzar y reforzar nuestra posición en los mercados y en el grado de compromiso de los clientes para con nuestro producto o servicio, sin tener que asumir grandes costes.

Como apoyo a esta estrategía de Marketing hemos mostrado por un lado en que redes se mueven los españoles, con el fin de dar a conocer a las pequeñas empresas los sitios donde deben hacer notar su presencia y por otro lado, detectamos la necesidad de contar con una herramienta que permita extraer las preferencias de los usuarios a partir de sus comentarios en estas plataformas, realizando así campañas controladas de captación de clientes para las Pymes, que permita atraerlos a sus negocios y conseguir pasar de un modelo social a un modelo económico.

Por último señalar que queda pendiente la validación de la estrategía planteada en una pequeña empresa.

[2] Retorno de Invesión: como de rentable está siendo nuestra introducción en redes sociales.

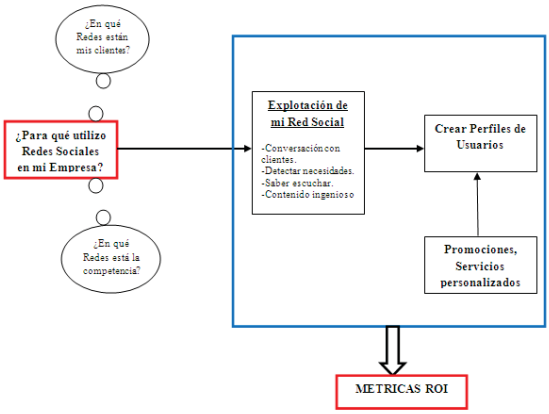


Figura 6. Estrategia de Marketing de ingreso de la pequeña empresa en las Redes Sociales

Referencias

1. Elogia Marketing Singular; iab Spain Research: *I Estudio sobre Redes Sociales en Internet*. En: http://www.elogia.net (2008).
2. Elogia Marketing Singular; iab Spain Research: *II Estudio sobre Redes Sociales en Internet*. En: http://www.elogia.net (2009).
3. Elogia Marketing Singular; iab Spain Research: *III Estudio sobre Redes Sociales en Internet*. En: http://www.elogia.net (2010).
4. Gómez, A.; Otero, C.: *Redes Sociales en la Empresa. La revolución e impacto a nivel empresarial y profesional.* Editorial RA-MA. (2011).
5. Liberos, E.: *Las redes sociales en España en 2011,* Social Media Marketing. (2011).
6. Martín-Albo, J.; Calero, C.; Moraga, Mª Angeles: *The Evolution of Social Networks: studying the past imagining the future,* enviado a IEEE Computing (Junio 2012)
7. O'Reilly,T. *What is Web 2.0. Design patterns and business models for the next generationofsoftware.*En: http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/whay-is-web-20.html. (2005).
8. Roca, G.: *Empresa en tiempso de redes. Retos y oportunidades.* Congreso Empresa 2.0 y Social Business. (2012)
9. Rull, L.: *Las mejores prácticasa de RRHH 2.0. Casos reales de éxito de empresas españolas.* Congreso Empresa 2.0 y Social Business. (2012).
10. Thompson, H.: *5 social media metrics you should be measuring*. http://socialmediatoday.com/heidisendible/329697/5-social-media-metrics-you-should-be-measuring (2011).

S^m4RIA Extension for OIDE: Desarrollo de Rich Internet Applications en la Web Semántica

Jesús M. Hermida, Santiago Meliá, Andrés Montoyo, Jaime Gómez

Departamento de Lenguajes y Sistemas Informáticos, Universidad de Alicante
Apartado de Correos 99, E-03080 Alicante, Spain
{jhermida, santi, montoyo, jgomez}@dlsi.ua.es

Resumen. El presente artículo describe la extensión S^m4RIA para la herrarmienta OIDE (*OOH4RIA Integrated Development Environment*), que implementa la metodología S^m4RIA en dicha herramienta . La aplicación, basada en el entorno Eclipse, soporta el desarrollo de los modelos S^m4RIA y los procesos de transformación (modelo a modelo y modelo a texto) que facilitan la generación de una aplicación RIA semántica, la cual puede compartir datos en forma de Linked Data y puede consumir datos desde la red de Linked Data. Además, de forma complementaria a la aproximación S^m4RIA original, la herramienta incluye mecanismos para la generación de interfaces RIA a partir de ontologías y para la generación automática de vistas de administración de las aplicaciones diseñadas.

1 Introducción

Las *Rich Internet Applications* (RIAs) han supuesto una mejora cualitativa en las interfaces de usuario aumentado la interoperabilidad de los componentes gráficos mediante un paradigma orientado a eventos, proporcionando una apariencia y funcionalidad similar a las aplicaciones de escritorio. No obstante, debido a cuestiones tecnológicas, las RIAs actúan como cajas negras que muestran el contenido de una forma amigable pero dificultan el acceso a los datos para algunos tipos de clientes Web (que requieren accesibilidad) como, por ejemplo, los buscadores Web. Este hecho ocurren tanto en las RIAs orientadas al navegador (cuyos datos se muestran en función de los eventos que los usuarios lanzan en la interfaz) o a los *plugins* (que son objetos binarios cuya información es únicamente posible visualizar mediante una extensión del navegador específica para cada tecnología de implementación). En este contexto, la aproximación S^m4RIA [1] (en inglés *Semantic Models for RIA*, modelos semánticos para RIA) presenta una metodología de diseño basado en modelos que extiende OOH4RIA y que permite el diseño y generación de RIAs semánticas (SRIA), que mejoran la interconexión y el intercambio de datos entre RIAs y con servicios externos haciendo uso de técnicas y tecnologías de la Web semántica.

El presente artículo describe las principales características de la extensión *S^m4RIA Extension for OIDE*, que implementa la aproximación S^m4RIA en OIDE [2] (*OOH4RIA Integrated Development Environment*). La extensión de OIDE, una apli-

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 123-126, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

cación basada en el entorno Eclipse, soporta y ayuda al usuario en el diseño de los modelos S^{m}4RIA. Asimismo, incluye los procesos de transformación (modelo a modelo y modelo a texto) necesarios para la generación de una aplicación RIA semántica, que comparte datos en forma de *Linked Data* (*http://linkeddata.org/*) y puede consumir datos desde la red de *Linked Data*. Además, de forma complementaria a la aproximación S^{m}4RIA original, la herramienta incluye mecanismos para la generación de interfaces RIA a partir de ontologías y para la generación automática de vistas de administración de las aplicaciones diseñadas.

Para contextualizar el trabajo, la siguiente sección presenta brevemente el método S^{m}4RIA y sus actividades. Para más información, Hermida et al. [1] describen la aproximación en detalle. Además la página web ***http://suma2.dlsi.ua.es/ooh4ria/sm4ria.html*** contiene información general y videos demostrativos.

2 El proceso de desarrollo S^{m}4RIA

La metodología S^{m}4RIA extiende el proceso original OOH4RIA modificando las tareas existentes e introduciendo una colección de nuevas tareas. El proceso de desarrollo se divide en tres actividades: 1) diseñar los componentes de la parte servidora de la SRIA, 2) diseñar los componentes de la parte cliente de la SRIA y, por último, 3) generar la SRIA por medio de un conjunto de transformaciones modelo a texto.

La primera actividad comienza cuando el diseñador del servidor define el modelo de Dominio, que incluye la información necesaria para especificar el diseño de la parte servidora de una RIA. A partir de este modelo, el diseñador de ontologías define la ontología de dominio alineando los conceptos incluidos en la SRIA con los conceptos de otras fuentes o aplicaciones que usen ontologías. Como resultado de esta tarea, se obtiene el modelo de dominio extendido (EDM, *Extended Domain Model*), que es, a su vez, un prerrequisito para definir Modelo de Navegación Extendido. En esta tarea, el diseñador define qué datos e instancias de la ontología del propio sistema SRIA van a ser accesibles por el resto de aplicaciones. Además, con modelo de Navegación Extendido los diseñadores pueden especificar el acceso a bases de conocimiento externas, utilizando consultas SPARQL, y cómo utilizar la información extraída.

La segunda actividad continúa con la transformación del modelo de navegación extendido en el modelo de Presentación y en el modelo de Orquestación en dos pasos consecutivos por medio de dos transformaciones llamadas *Nav2Pres* y *Nav-Pres2Orch*. El modelo de Presentación representa la estructura de los elementos de la interfaz de la SRIA que es completada por el modelo de Orquestación, que representa el comportamiento de la interfaz definida en el modelo de Presentación.

Finalmente, en la última actividad del método se genera la SRIA a partir del conjunto de modelos creados en las dos primeras actividades por medio de un conjunto de procesos de transformación modelo a texto.

3 *S^{m}4RIA Extension for OIDE*: Principales características

OIDE es una aplicación basada en el entorno Eclipse, desarrollada como un conjunto de extensiones (plug-ins) del mismo, que soporta la metodología OOH4RIA para el desarrollo de RIAs. Concretamente la aplicación define los metamodelos

OOH4RIA en formato Ecore y utilizando la plataforma EMF/GMF permite definir de forma gráfica los cuatro modelos de OOH4RIA: dominio, navegación y presentación-orquestación. Asimismo, la herramienta da soporte al proceso de generación de código que permite obtener la mayor parte de los componentes (tanto cliente como servidor) de una aplicación RIA. Para ello implementa un conjunto de reglas Xpand que transforman los modelos en código C# utilizando los *frameworks* de Silverlight y WCF (*Windows Communication Foundation*).

Sobre la base de OIDE, *S^m4RIA extension for OIDE* implementa los artefactos y procesos de la metodología S^m4RIA como una nueva funcionalidad de Eclipse. Este apartado describe los elementos incorporados y modificados en la herramienta original que facilitan el desarrollo del proceso descrito en S^m4RIA y la generación automática de la mayor parte de los componentes software de una SRIA. En concreto, la extensión S^m4RIA para OIDE incluye las siguientes funcionalidades y componentes:

a) **Nuevos modelos.** Utilizando las bibliotecas EMF y GMF se han desarrollado tres nuevos modelos, cuyos meta-modelos han sido definidos a partir del meta-modelo Ecore:
   - *Modelo de dominio extendido*: es un nuevo modelo que permite modelar ontologías ligeras en la herramienta y mapear los elementos del modelo de dominio en elementos de ontología.
   - *Modelo de navegación extendido*: es una extensión del modelo de Navegación de OOH4RIA que permite definir clases navegacionales a partir del modelo de dominio extendido y enlaces de navegación a servicios externos, que pueden ser combinados creando *mashups*. En un principio se ha implementado el acceso a los principales servicios de Linked Data, llamados puntos de acceso de SPARQL.
   - *Modelo ontológico de visualización*: es un nuevo modelo que permite representar características tanto de estructura como de comportamiento de la interfaz de usuario desde el punto de vista del usuario (en contraposición a la visión que tiene el diseñador).

b) **Transformaciones modelo a modelo.** Se han definido un conjunto de reglas de transformación que facilitan y aceleran el desarrollo del método de diseño. Para la especificación y ejecución de las reglas se ha utilizado el lenguaje y el motor de reglas *QVT operational*. Las transformaciones definidas en la extensión son las siguientes (las transformaciones M_a – M_b son unidireccionales, es decir, transforman M_a en M_b):
   - *Transformación Dom2DomExt*: Modelo de Dominio – Modelo extendido de dominio. Esta transformación permite obtener una ontología a partir del modelo de dominio. A partir del EDM generado, el diseñador puede adaptarlo a sus necesidades.
   - *Transformación DomExt2NavExt*: Modelo Extendido de dominio – Modelo extendido de navegación. A partir del modelo extendido de dominio esta transformación obtiene la vista del modelo de navegación extendido para agentes software.
   - *Transformación NavExt2Pres&Orch*: Modelo Extendido de navegación – Modelo de presentación. Esta transformación implementa las transformaciones

Nav2Pres y *Nav&Pres2Orch*, creando una presentación predeterminada a partir del modelo de navegación.

c) **Transformaciones modelo a texto.** Para la generación del código fuente de la RIA semántica se han definido una serie de reglas de transformación Xpand, que han sido agrupadas en función de los módulos de la SRIA que generan:
   - *Generación de ontologías OWL y reglas de mapeo*: Este bloque de reglas permite generar la ontología de dominio y las reglas de mapeo necesarias para generar Linked Data utilizando un conversor de base de datos a RDF. Además permite generar las ontologías de navegación y visualización, que dan una visión global de la RIA semántica.
   - *Generación de los módulos de acceso a Linked Data*: Este bloque genera los componentes necesarios para acceder a los puntos de acceso SPARQL y recuperar Linked Data.
   - *Generación de los módulos para el acceso a los datos locales*: Este bloque genera una interfaz HTML para la RIA, que es procesable por aquellos clientes que no tienen acceso a la interfaz Silverlight.

d) **Nuevos procesos de modernización y reingeniería.** Los nuevos elementos implementados facilitan la adaptación de la metodología S^m4RIA a nuevas procesos de generación. Entre ellos, podemos destacar las siguientes :
   - *Generación de vistas RIA de fuentes de Linked Data*. Por medio de dos transformaciones M2M que generan un modelo de Dominio y un EDM a partir de una ontología OWL, es posible definir un servidor RIA que gestione los datos de un servicio de Linked Data dando una vista RIA a los datos contenidos en el repositorio de Linked Data.
   - *Generación automática de vistas de administración de aplicaciones*. Utilizando las transformaciones M2M definidas es posible generar automáticamente, con un esfuerzo muy reducido para el diseñador, vistas de administración de las aplicaciones RIA modeladas a partir de su modelo de Dominio.

Las capturas de pantalla y los videos de la herramienta desarrollada están disponibles en el sitio Web de OOH4RIA–S^m4RIA: *http://suma2.dlsi.ua.es/ooh4ria/sm4ria.html*

Agradecimientos

El presente artículo ha sido financiado por el Ministerio de Educación, Cultura y Deporte del Gobierno de España bajo el programa FPU (AP2007-03076) y el proyecto SONRIA (TIN2010-15789). Asimismo, los autores agradecen el soporte económico de la Universidad de Alicante mediante el proyecto de investigación DIMENRIA (GRE10-23).

Referencias

1. Hermida, J.M., Meliá, S., Montoyo, A., Gómez, J.: Developing Rich Internet Applications as Social Sites on the Semantic Web: A Model-Driven Approach. IJSSOE 2(4), 21-41, 2011.
2. Meliá, S., Martinez, J., Mira, S., Osuna, J., Gómez, J.: An Eclipse Plug-in for Model-Driven Development of Rich Internet Applications. En actas del ICWE 2008. LCNS 6189, pp. 514-517 (2010)

TOUCHE CASE Tool: A Task-Oriented and User-Centered Case Tool to Develop Groupware Applications

Víctor M. R. Penichet, María D. Lozano, José A. Gallud, Ricardo Tesoriero

Universidad de Castilla-La Mancha. Departamento de Sistemas Informáticos
Av. España S/N, Campus Universitario de Albacete, (02071), Albacete, España
victor.penichet@uclm.es, maria.lozano@uclm.es,
jose.gallud@uclm.es, ricardo.tesoriero@uclm.es

Resumen
TOUCHE es un modelo de proceso y una metodología para el desarrollo de interfaces de usuario para aplicaciones groupware desde la elicitación de requisitos hasta su implementación, considerando las características y particularidades de estos sistemas colaborativos desde el inicio. El estudio de los conceptos específicos ha concluido con un modelo conceptual de términos y relaciones sobre el que se asienta el modelo de proceso propuesto. Para soportar este proceso se presenta TOUCHE CASE Tool, una herramienta CASE que permite automatizar dicho proceso de desarrollo.

1 Introducción

Los modelos de proceso y metodologías propuestos tradicionalmente en la Ingeniería del Software muestran su potencia real cuando hay una herramienta CASE que les da el soporte necesario para llevar a cabo los proyectos desde el inicio, automatizando ciertas tareas que hacen más sencilla la especificación del sistema. Estas herramientas CASE ayudan a mantener la coherencia del sistema, pero sobre todo realizan de forma automática un conjunto de acciones que de modo manual serían repetitivas, muy laboriosas y podrían llevar a confusión. Esta automatización disminuye el trabajo de los analistas y demás participantes en el proceso de desarrollo de la aplicación CSCW, generando código desde la misma etapa de elicitación de requisitos.

TOUCHE [1,2] es un modelo de proceso y una metodología para el desarrollo de interfaces de usuario para aplicaciones groupware desde la elicitación de requisitos hasta su implementación, considerando las características y particularidades de estos sistemas colaborativos desde el inicio. Para dar soporte a este modelo de proceso y automatizar la definición de los distintos artefactos involucrados, se ha implementado una herramienta CASE, denominada TOUCHE CASE Tool que permite hacer un seguimiento guiado del modelo de proceso, reutilizar elementos durante todo el proceso y asegurar y mantener la trazabilidad desde las etapas iniciales hasta el final.

En este trabajo se presenta la herramienta TOUCHE CASE Tool aplicada a un caso concreto para mejorar la comprensión de sus características.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 127-130, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

2 TOUCHE CASE Tool

TOUCHE CASE Tool (Task-Oriented and User-Centred Process Model for Developing Interfaces for Human-Computer-Human Environments Computer-Aided Software Engineering Tool) es la herramienta CASE que soporta el modelo de proceso y la metodología propuestos en TOUCHE. Esta herramienta asiste al usuario en el desarrollo de interfaces de usuario centrado en el usuario y dirigido por tareas para sistemas CSCW.

Por su naturaleza CASE, TOUCHE CASE Tool, facilita la automatización de algunos procesos en el ciclo de vida del software, relaciona y muestra la información desde diferentes perspectivas facilitando al desarrollador esta ardua tarea y mantiene la coherencia del desarrollo al garantizar la trazabilidad inter- e intraetapa entre los diferentes modelos.

2.1 Elicitación de Requisitos

Los datos y plantillas elaborados en esta etapa se encuentran bajo el primer nodo del proyecto, como se muestra en la Figura 1. Seleccionando el nodo de la etapa de requisitos se pueden insertar los datos relacionados con la *introducción* y la *descripción del sistema* a desarrollar. En esta etapa, la herramienta permite además, crear los diagramas de Casos de Uso y su correspondiente trazabilidad al resto de las etapas. Se generan automáticamente a partir de los datos obtenidos acerca de los participantes del sistema y los requisitos funcionales capturados en las plantillas.

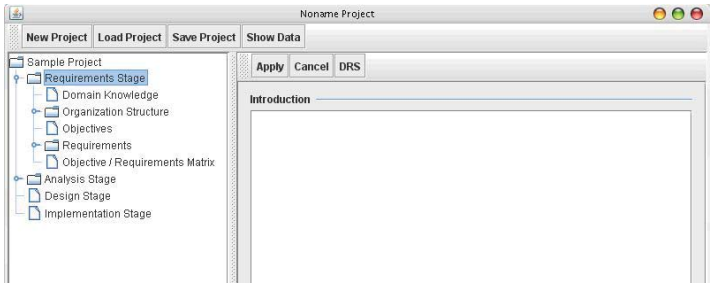


Figura 1. Detalle de la pantalla principal en la *Elicitación de Requisitos*

2.2 Análisis, Diseño e Implementación

La etapa de análisis está compuesta básicamente por dos elementos principales: las tareas y los roles. En el árbol de la izquierda, se muestran los datos organizados de la siguiente manera: roles, dónde cada rol muestra los autores, las fuentes, responsabilidades, capacidades y el diagrama CTT de la asociación de las tareas con los roles. La Figura 2 muestra una plantilla con el resumen de datos de un rol.

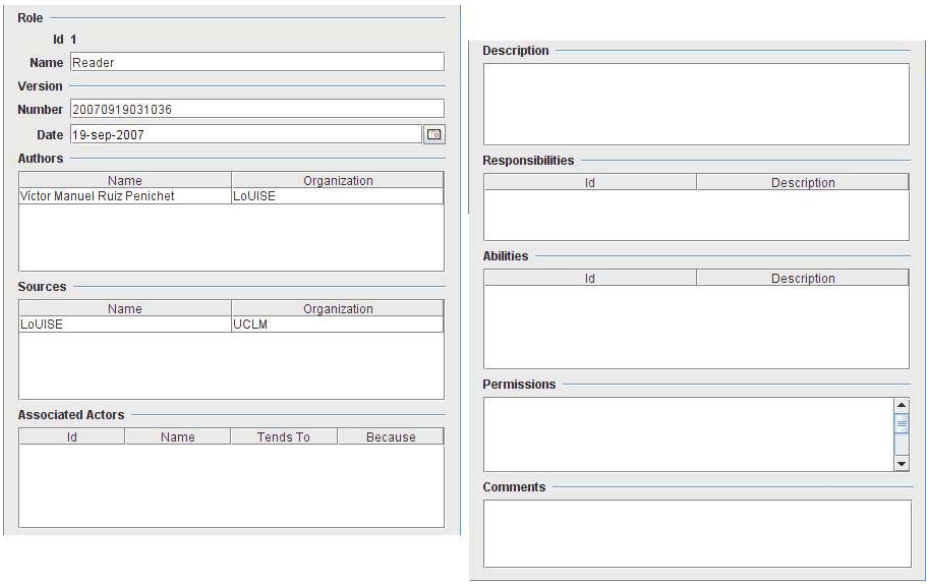


Figura 2. Información relativa a la definición de un rol determinado

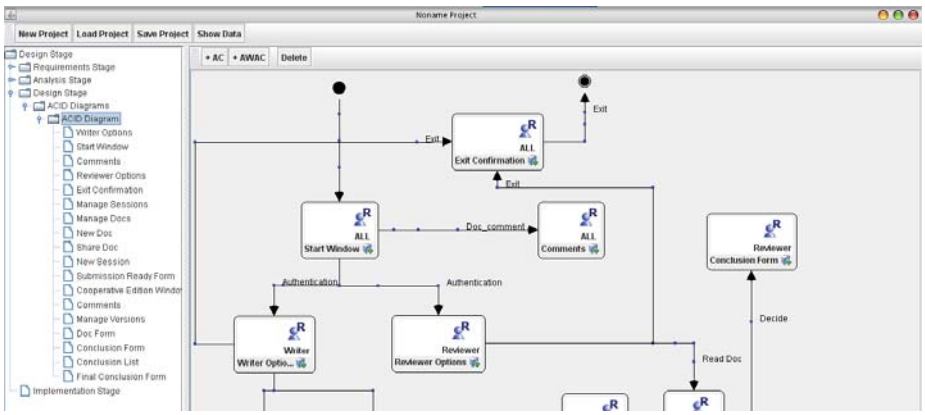


Figura 3. Detalle de un ACID

La Figura 3 y la Figura 4 muestran los diagramas relativos a la etapa de diseño: Diagrama de Interacción de Contenedores Abstractos (ACID) y Diagrama de Interfaces de Usuario Abstractas (AUID) respectivamente.

El ACID permite modelar la navegación entre los diferentes contenedores, a menudo ventanas, de la aplicación según el rol que se desempeñe y según qué tarea se lleve a cabo.

El AUID modela cómo sería cada uno de esos contenedores teniendo en cuenta, además de los aspectos relativos a HCI según los trabajos previos, aspectos típicos de las aplicaciones groupware que favorecen la percepción de los usuarios del grupo.

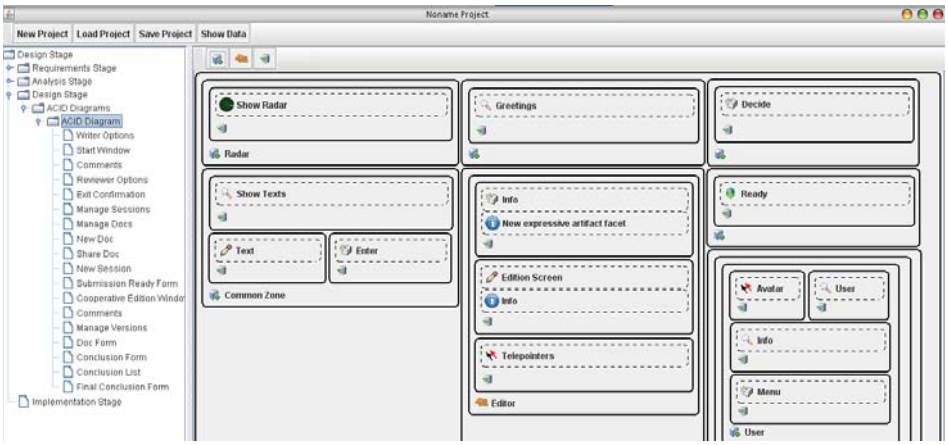


Figura 4. Detalle de un AUID

3 Conclusiones

En este trabajo se presenta la herramienta TOUCHE CASE Tool, una herramienta CASE diseñada para dar soporte al modelo de proceso seguido por TOUCHE definido para la generación de interfaces de usuario para aplicaciones groupware.

La herramienta guía al desarrollador a través de las distintas fases del proceso de desarrollo asistiendo en la generación de los distintos diagramas y automatizando la trazabilidad requerida en cada etapa. Más información detallada de la herramienta, así como la herramienta en sí, se puede encontrar en www.penichet.net.

4 Agradecimientos

Esta publicación ha sido financiada por los proyectos nacionales: CENIT-2008-1019 y CICYT-TIN 2011-27767-C02-01. Así como por los proyectos regionales de la JCCM: PPII10-0300-4174 y PII2C09-0185-1030.

Referencias

1. Victor M. R. Penichet, María D. Lozano, José A. Gallud, Ricardo Tesoriero: User Interface Analysis for Groupware Applications in the TOUCHE Process Model. ISSN: 0965-9978. 40, 12 (Dec. 2009), 1212-1222. International Journal Advances in Engineering Software (ADES), 2009
2. Víctor M. R. Penichet, María D. Lozano, José A. Gallud, Ricardo Tesoriero: Requirement-based Approach for Groupware Environments Design. Journal of Systems and Software (JSS), 2010. ISSN: 0164-1212. DOI: 10.1016/j.jss.2010.03.029

CSRML Tool: una Herramienta para el Modelado de Requisitos de Sistemas Colaborativos

Miguel A. Teruel, Elena Navarro, Víctor López-Jaquero, Francisco Montero, Pascual González

LoUISE, Instituto de Investigación en Informática, Universidad de Castilla - La Mancha
{miguel, enavarro, victor, fmontero, pgonzalez}@dsi.uclm.es

Resumen. Cada vez más aplicaciones incluyen algún tipo de soporte a la realización de tareas colaborativas. Como para cualquier otro tipo de sistema, una especificación de requisitos precisa es uno de los pilares básicos en el desarrollo de este tipo de sistemas con soporte a la colaboración. Sin embargo, este tipo de sistemas poseen un tipo especial de requisitos difícil de especificar con las técnicas de Ingeniería de Requisitos (IR) tradicionales. Estudios experimentales previos muestran que la especificación del conocimiento o la percepción de una situación o hecho (*awareness*), necesarios para que los usuarios puedan llevar a cabo tareas colaborativas, puede ser excesivamente compleja, o incluso incompleta, cuando se usan la técnicas clásicas de IR. Para solventar este problema, se ha desarrollado CSRML (*Collaborative Systems Requirements Modeling Language*), una extensión del lenguaje orientado a objetivos *i** para especificar requisitos de sistemas colaborativos. En este trabajo se presenta CSRML Tool: una herramienta que da soporte al lenguaje CSRML. Gracias a esta herramienta, se podrán modelar los distintos diagramas que conforman la especificación de los requisitos de un sistema colaborativo utilizando CSRML, así como comprobar la validez de los mismos con respecto al meta-modelo de CSRML.

Palabras clave: ingeniería de requisitos; sistemas colaborativos; workspace awareness; Visualization and Modeling SDK; Visual Studio; CSRML.

1 Introducción

Durante los últimos años, la forma en la que Internet proporciona servicios, aplicaciones, etc. ha cambiado notablemente. Hoy en día, la colaboración está en todas partes. De hecho, si observamos el ranking de las páginas web más visitadas, prácticamente las 100 primeras son webs colaborativas [1]. Redes sociales, editores de texto multiusuario, juegos online… todo tiende a ser colaborativo. Un buen ejemplo para entender cómo funciona la colaboración entre usuarios es Google Docs [2], una aplicación web cada vez más utilizada, que permite a varios usuarios editar un documento de texto de forma simultánea. Este tipo de herramienta es un interesante ejemplo de sistema CSCW (*Computer Supported Cooperative Work*) [3,

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 131-144, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

4]. Gracias a estos sistemas, junto a las funcionalidades que ya proporcionaban las aplicaciones software clásicas, los usuarios pueden realizar tareas relacionadas con colaboración, comunicación y coordinación, también conocidas como tareas *3C*. No obstante, el principal problema cuando se definen los requisitos de un sistema CSCW no es únicamente la especificación de esas tareas 3C. Existen otros requisitos, de naturaleza altamente no funcional, éstos surgen de la necesidad que tienen los usuarios de ser, por ejemplo, conscientes de la presencia o ausencia de otros usuarios con los que realizar las anteriormente mencionadas tareas 3C, siendo un ejemplo de ello el denominado *Workspace Awareness* (WA) [5]. WA incluye conocimiento acerca de, por ejemplo, *quién* hay disponible para colaborar, *qué* están haciendo (o hicieron en el pasado) el resto de usuarios, *cuándo* un artefacto fue modificado o *cómo* ocurrió alguna operación.

Estudios experimentales previos [6, 7] muestran que la especificación del conocimiento o la percepción de una situación o hecho (*awareness*), necesarios para que los usuarios puedan llevar a cabo tareas colaborativas, puede ser excesivamente compleja, o incluso incompleta, cuando se usan la técnicas clásicas de IR. Ello ha motivado la definición del lenguaje CSRML [8] como una extensión del lenguaje orientado a objetivos *i** [9, 10]. No obstante, CSRML no disponía de una herramienta de modelado, haciendo que la especificación de sistemas fuera compleja y sin soporte automático para verificar los modelos creados.

En el contexto de las técnicas orientadas a objetivos una de las herramientas más conocidas es OpenOME [11] que da soporte entre otras metodologías, a *i** [9, 10] en la que se basa CSRML (véase Sección 2). Ésta es una herramienta orientada a objetivos y a agentes que soporta tanto el modelado como el análisis de requisitos. Es de destacar que esta herramienta dispone de dos versiones: aplicación desktop y plug-in para Eclipse [12]. Podemos encontrar otras herramientas que dan soporte a *i** (o a otras metodologías orientadas a objetivos) como por ejemplo IStarTool [13], DesCARTES [14] o J-PRiM [15], pero hasta donde nosotros sabemos, ninguna de ellas ofrece la flexibilidad necesaria para incorporar los elementos expresivos de CSRML. Con el fin de solventar este problema surge la herramienta *CSRML Tool*. En este trabajo se presenta dicha herramienta que da soporte a CSRML permitiendo tanto el modelado de sistemas colaborativos a través de un conjunto de diagramas como la validación automática de los mismos.

Este artículo se estructura de la siguiente manera: después de esta introducción, en la sección 2 se presenta el lenguaje CSRML para, seguidamente, en la sección 3 presentar la herramienta que da soporte al mismo, CSRML Tool, mostrando detalles sobre su implementación e interfaz, así como describiendo su funcionamiento mediante un caso de estudio. Finalmente, la sección 4 documenta nuestras principales conclusiones e identifica trabajos futuros.

2 El Lenguaje CSRML

El lenguaje CSRML (*Collaborative Systems Requirements Modeling Language*) es un lenguaje de IR Goal-Oriented (GO) [16], basado en *i** [9, 10]. CSRML se propone con el objetivo de facilitar la especificación de sistemas colaborativos en los cuales la

definición de ciertos requisitos ligados a la colaboración entre usuarios es difícil (o incluso imposible) de representar con las técnicas clásicas de IR. Es importante señalar que la definición de CSRML, así como de la herramienta que da soporte al mismo, no pueden considerarse trabajos aislados, sino que han sido definidos por un proceso guiado por evaluaciones empíricas:

- En un trabajo inicial [6], se analizaron tres técnicas de IR (Use Cases [17, 18], Viewpoints [19] y Goal-Oriented [16]) con el fin de identificar cuál de ellas era la más adecuada para especificar los requisitos especialmente necesarios para la descripción de un sistema colaborativo, concluyendo que Goal-Oriented era la técnica más adecuada para este fin.
- A continuación, en [7], se realizó un análisis de algunos de los enfoques Goal-Oriented más conocidos (NFR Framework [20], KAOS [16] e *i** [9, 10]) para averiguar cuál de ellos podía ser aplicado en sistemas colaborativos de la forma más adecuada. A pesar de que ninguno de estos enfoques satisfizo todas las necesidades de expresividad requeridas, la notación *i** fue la mejor posicionada para la realización de esta tarea.
- El resultado anterior nos condujo a justificar la propuesta de CSRML [8, 21] como una extensión de *i** que resolvía los problemas de expresividad de este último. Entre esas limitaciones cabe destacar la representación de la colaboración y el *awareness* entre usuarios, así como la gestión de actores / roles.
- Finalmente, esta primera versión de CSRML fue evaluada mediante una familia de experimentos [22, 23] que pese a demostrar que CSRML era adecuado para el modelado de las distintas características de un sistema colaborativo, nos hizo descubrir ciertos problemas en la representación del *awareness* y de la importancia entre tareas colaborativas, lo que condujo al desarrollo de la segunda versión de CSRML, la cual solventa estos problemas mediante la inclusión / modificación de nuevos elementos expresivos.

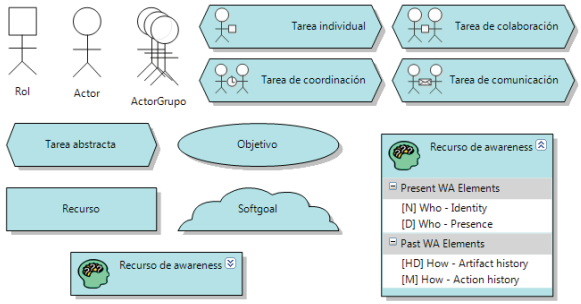


Fig. 1. Elementos de CSRML

Una vez vistos los trabajos que condujeron a CSRML, y antes de presentar la herramienta que da soporte al mismo, haremos un recorrido por los distintos elementos que forman parte de este lenguaje (Fig. 1):

- Un *rol* (*role*) representa el conjunto de conocimientos, destrezas y habilidades que debe poseer un actor para desarrollar con competencia una serie de tareas. Un actor que representa un rol puede participar en tareas individuales y colaborativas

(mediante enlaces de participación) y puede ser responsable de la consecución de un objetivo (a través de enlaces de responsabilidad).

- Un *actor* es un usuario, programa o entidad con ciertas capacidades adquiridas que puede interpretar un rol responsable de ciertas acciones. Un actor tiene que representar un rol (lo cual se especifica mediante un enlace de representación) para participar en el sistema.
- Un *actorgrupo* (*groupactor*) representa un grupo compuesto de uno o más actores cuyo objetivo es lograr uno o más objetivos específicos del grupo.
- Una *tarea* (*task*) especifica una forma particular de hacer algo. La importancia de una tarea se define mediante un código de colores: verde (menos importante), amarillo, naranja, rojo (más importante). Además, CSRML identifica dos tipos especiales de tareas:
  - Tarea abstracta (*abstract task*): Este tipo de tarea es una abstracción de un conjunto de tareas concretas y, posiblemente, de otros elementos, como objetivos, recursos o *softgoals*. No pueden asignarse enlaces de participación directamente a este tipo de tareas.
  - Tarea concreta (*concrete task*): Estas son las tareas en las que están involucrados los participantes, los cuales serán asignados a estas tareas mediante los enlaces de participación. Las tareas abstractas se refinan en estas tareas. A su vez, se subdividen en cuatro tipos: (i) *tareas individuales*, que los actores pueden realizar sin ningún tipo de interacción con otros actores; (ii) *tareas de colaboración*; (iii) *tareas de comunicación*; y (iv) *tareas de* coordinación. Los últimos tres tipos responden a tareas en las que dos o más actores participan (denominadas tareas 3C).
- Un *objetivo* (*goal/hardgoal*) responde a las preguntas "¿qué desea alcanzar el usuario?" "¿a qué aspira el usuario?". Describe un cierto estado que un actor desearía alcanzar. No obstante, un objetivo no describe cómo debería ser alcanzado.
- Un *softgoal* es una condición que a un actor le gustaría conseguir, aunque al contrario que para el concepto de objetivo (*hardgoal*), la condición para conseguirlo no está claramente definida.
- Un *recurso* (*resource*) es una entidad (física o de información) que un actor necesita para conseguir un objetivo o realizar una tarea. El mayor interés sobre un recurso es si está disponible y para quién lo está.
- Un *recurso de awareness* (*awareness resource*) es una especialización del recurso que representa una necesidad de percepción que facilita y en algunos casos posibilita a un rol realizar una tarea proporcionándole el *feedback* requerido. Este elemento representa un conjunto de atributos relacionado con un *enlace de participación* entre un rol y una tarea. Sobre un diagrama, este elemento puede mostrarse de forma contraída o expandida (véase Fig. 1) para facilitar su legibilidad de las especificaciones y reducir la sobrecarga de información. En su forma expandida, este recurso muestra las características del *Workspace Awareness* identificadas por Gutwin [5] que pueden ser establecidas (si son necesarias) mediante el grado de contribución al cumplimiento de la tarea a la que el recurso

va asociado. La importancia puede ser *bueno* (N*, nice to have*), *deseable* (D, *desirable*)*, altamente deseable* (HD, *highly desirable*) u *obligatorio* (M, *mandatory*).

El conjunto anterior de elementos se relacionará entre sí mediante los siguientes enlaces (Fig. 2):

- Una *dependencia* (*dependency*) documenta una relación entre dos roles. Un rol depende de otro para conseguir un objetivo, realizar una tarea o usar un recurso.
- Un *means-end link* documenta qué *softgoals*, tareas y/o recursos contribuyen a conseguir un objetivo. Estos enlaces facilitan la documentación y la evaluación de las distintas formas de satisfacer un objetivo, por ejemplo, realizando varias descomposiciones de un objetivo en distintos *softgoals*, tareas y recursos.
- Un *enlace de descomposición de tareas* documenta los pasos necesarios para realizar una tarea. Este enlace relaciona una tarea con sus componentes, los cuales pueden ser cualquier combinación de sub-objetivos, sub-tareas, recursos o *softgoals*. La descomposición de una tarea abarca las sub-tareas que pueden realizarse, los sub-objetivos que deberían cumplirse, los recursos que se necesitan y los softgoals que típicamente definen objetivos de calidad para la tarea.
- Una *contribución* (*contribution*) documenta una influencia *positiva*, *negativa* o *desconocida* de ciertos *softgoals* sobre otros *softgoals* o tareas.
- Un *enlace de representación* (*playing link*) sirve para representar cuándo un actor asume un rol. Este enlace tiene una condición de guarda, que establece la condición que debe cumplirse para que el actor interprete el rol.
- Un *enlace de participación* (*participation link*) indica quién está involucrado en una tarea. Este enlace tiene un atributo para especificar su cardinalidad, es decir, el número de usuarios que pueden estar involucrados en dicha tarea. A este enlace podrá ir asociado un *recurso de awareness.*
- Un *enlace de responsabilidad* (*responsability link*) asigna un rol (interpretado por un actor) a un objetivo, *softgoal* o tarea. Este enlace representa quién es el *stakeholder* responsable del cumplimiento de una tarea u objetivo. No es necesario que un *stakeholder* esté involucrado en las sub-tareas del objetivo. No obstante, si el rol es responsable de una tarea u objetivo, este rol es también responsable de los elementos en los que éstos se dividen, a menos que un nuevo enlace de responsabilidad llegue a uno de estos elementos.

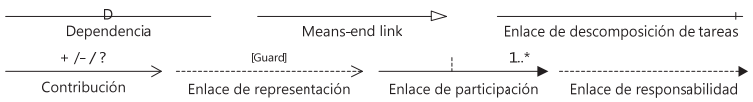


Fig. 2. Relaciones en CSRML v2

Finalmente, para estructurar una especificación de requisitos en CSRML de una forma ordenada y complementaria, se definen cinco tipos de diagramas. Algunos de ellos se usarán sólo una vez en la especificación del sistema, pero otros serán usados cuantas veces sean necesarios, dependiendo de las necesidades específicas del sistema. El conjunto completo de diagramas en CSRML y sus relaciones entre ellos se muestran en la siguiente figura (ver Fig. 3). Así, una especificación de requisitos en

CSRML se estructura por medio de los siguientes diagramas, que se detallan mediante un ejemplo en la sección 3.3:

- *Diagrama de jerarquía de grupo* (*Group Hierarchy Diagram, GHD*): éste es el primer diagrama que será empleado cuando se especifica un sistema colaborativo con CSRML (Fig. 8). En este diagrama se especificarán los grupos de *actores* que participarán en el sistema mediante los correspondientes *actorgrupo*. En un GHD se usan *enlaces de participación* para denotar qué actores forman cada grupo (con su correspondiente cardinalidad). Además, si un grupo tiene un líder (actor responsable de coordinar el grupo y ser el delegado en la interacción con otros grupos) su nombre se mostrará subrayado.

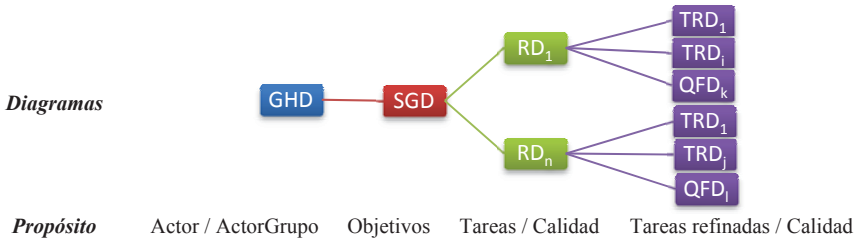


Fig. 3. Estructura de diagramas de CSRML

- *Diagrama de objetivos del sistema* (*System Goals Diagram, SGD*): en este diagrama (Fig. 9) se especifican los objetivos de actores y grupos de actores, así como la tarea principal del (sub)sistema. Estos actores y grupos de actores tendrán una cardinalidad (entre corchetes) que representará cuantas ocurrencias de los mismos participarán en el cumplimiento de un objetivo.
- *Diagramas de responsabilidades* (*Responsibility Diagrams, RDs*): en este diagrama (Fig. 10) se representa la descomposición de la tarea principal identificada en el diagrama anterior. Además, se usarán *enlaces de responsabilidad* para denotar qué actores (representando un rol) son responsables de cada objetivo, tarea o softgoal. Cada tarea representada en este diagrama se refinará por medio de un *diagrama de refinamiento de tareas.*
- *Diagramas de refinamiento de tareas* (*Task Refinement Diagrams, TRDs*): en estos diagramas (Fig. 11) se refinan las tareas identificadas en el RD en tareas más concretas y/o nuevos objetivos hasta que se especifiquen tareas concretas (individuales o colaborativas). Pueden haber tantos TRD como sean necesarios.
- *Diagramas de factores de calidad* (*Quality factors diagrams, QFDs*): en este último tipo de diagrama (Fig. 12) se especifican los factores de calidad que contribuyen a lograr los principales softgoals (factores de calidad) identificados en los RD. Estos softgoals se relacionan con el softgoal principal de calidad mediante *contribuciones.*

3 CSRML Tool: dando soporte al lenguaje CSRML

Una vez presentados los principales elementos del lenguaje CSRML, se procede a la presentación de una herramienta CASE que proporciona soporte al mismo. Para su

implementación se decidió la utilización de *Microsoft Visualization and Modeling SDK* (VMSDK, anteriormente conocido como DSL tools) [24], que permite crear herramientas de desarrollo basadas en modelos integradas con Visual Studio, entorno de desarrollo ampliamente utilizado de Microsoft.

Esta plataforma de creación de DSLs (*Domain Specific Languages*) de Microsoft fue seleccionada para crear CSRML Tool por su potencia gráfica necesaria para representar los elementos de CSRML (bastante complejos en algunos casos como el de los recursos de *awareness*), así como por la experiencia previa del equipo de desarrollo de la aplicación en entornos Microsoft .NET, entre otras razones. Además, dada su integración con Visual Studio, esta herramienta resulta fácil de utilizar por cualquier usuario con un mínimo de experiencia en este entorno. Dadas las facilidades que proporciona el entorno, las características de CSRML Tool son las siguientes:

- Especificación completa de un sistema colaborativo en CSRML mediante el uso de los distintos diagramas que componen el mismo.
- Validación automática de modelos respecto al meta-modelo de CSRML (Fig. 4 y Fig. 5), impidiendo la especificación de modelos no consistentes. Esta validación se lleva a cabo tanto mediante las restricciones establecidas en el metamodelo como utilizando el motor de reglas proporcionado por VMSDK.
- Interfaz de usuario sencilla integrada dentro de Visual Studio, con distintas áreas de trabajo como explorador de modelos o los cuadros de herramientas para los distintos diagramas de CSRML (Fig. 7)
- Generación de código para el enlace de CSRML con otras metodologías para el desarrollo de interfaces de sistemas colaborativos [25] (actualmente en desarrollo).
- Integración con Windows y Visual Studio mediante un sencillo programa instalador el cual integra los ficheros con extensión **.csrml* dentro del sistema operativo.

En el resto de esta sección se documentan detalles de esta herramienta respectivos a su implementación (Sección 3.1) e interfaz de usuario (Sección 3.2). Finalmente, se utilizará la herramienta para el modelado de un sistema colaborativo basado en una actividad de e-learning multiusuario cooperativa (Sección 3.3).

3.1 Detalles de implementación de CSRML Tool

Sin ahondar en exceso en los detalles de implementación de esta herramienta, en esta sección se hará una breve presentación a nivel gráfico a través de su meta-modelo (Fig. 4 y Fig. 5). El principal problema que se identificó inicialmente, es que VMSDK no usa un sistema de meta-modelado basado en UML, sino que usa un sistema propietario mediante el que, si bien resulta más sencillo el desarrollo del DSL, fue necesaria una conversión de nuestro meta-modelo original. Véase como ejemplo la parte gráfica de la definición de un recurso de *awareness* (Fig. 6).

Tras la definición del meta-modelo dentro de VMSDK, se procedió a la definición de los elementos gráficos de CSRML más complejos (tareas, softgoals, recursos de awareness), dado que este entorno sólo permite la inclusión de imágenes estáticas, además de cuatro formas geométricas básicas. Así, estos elementos complejos han

sido implementados mediante los servicios de *.net printing* [26], permitiendo de esta forma la inclusión de elementos no soportados de forma nativa por el entorno.

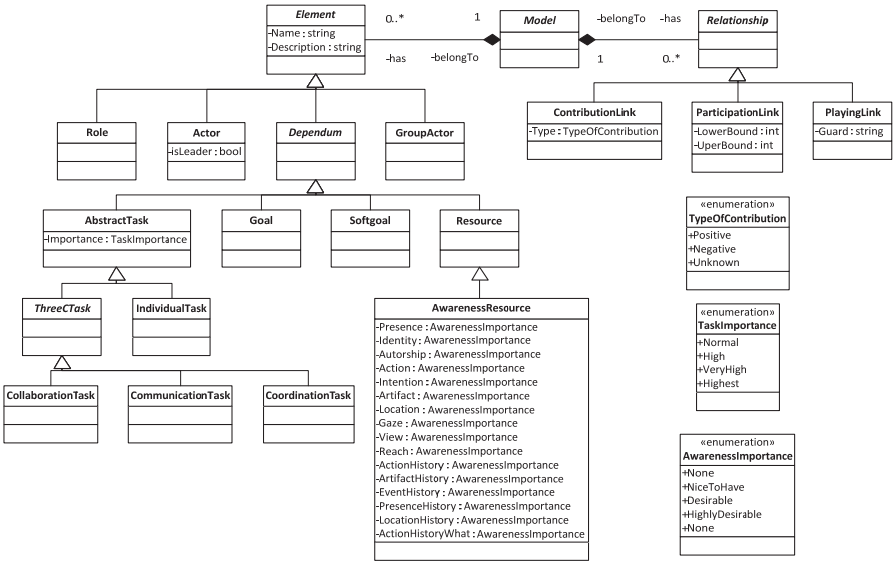


Fig. 4. Meta-modelo de CSRML (primera parte)

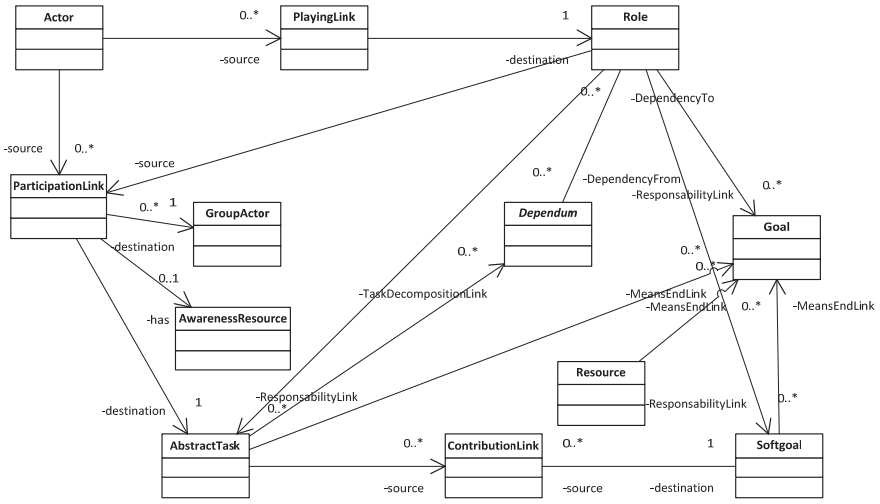


Fig. 5. Meta-modelo de CSRML (segunda parte)

Una vez que CSRML Tool soporta todos los elementos gráficos de CSRML, se procedió al desarrollo del código correspondiente con las restricciones necesarias para evitar todo tipo de inconsistencias en los modelos. Por ejemplo, cuando el usuario intente guardar un modelo en que más de un rol participe en una tarea individual, o bien exista algún tipo de recursividad en la descomposición de alguna tarea, el entorno advertirá al usuario y le indicará dónde está el error dentro del modelo.

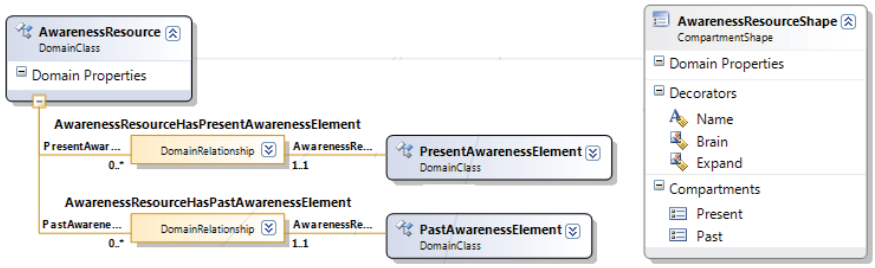


Fig. 6. Vista parcial del meta-modelo implementado dentro de VMSDK

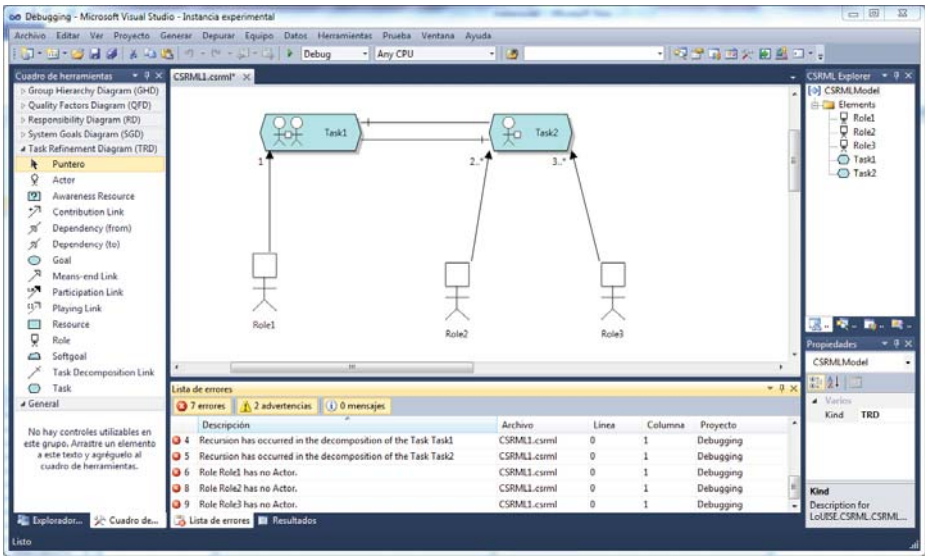


Fig. 7. Interfaz de usuario de CSRML Tool

Por último, se procedió a la mejora de la interfaz de usuario, creando el explorador de modelos CSRML en el cual el usuario puede navegar dentro una estructura en forma de árbol. Además, se crearon distintas cajas de herramientas para facilitar a los usuarios la definición de los distintos tipos de diagramas. Finalmente, también se creó un programa instalador que integra de forma automática CSRML con Windows y Visual Studio.

3.2 La interfaz de usuario de CSRML Tool

Una vez comentados distintos detalles relativos a la implementación de CSRML Tool, en esta sección se describe su interfaz de usuario (véase Fig. 7) en la que podemos contemplar la edición de un modelo con errores de validación.

La interfaz de CSRML Tool cuenta con los siguientes apartados:

- Área de modelado (parte superior central): permite editar y validar modelos.
- Cuadro de herramientas (parte izquierda): contiene todos los elementos y relaciones de CSRML agrupados por tipo de diagrama.

- Explorador del modelo (parte superior derecha): facilita la navegación y edición de un modelo dentro de una estructura con forma de árbol.
- Lista de errores de validación (parte inferior central): muestra los errores cometidos en el modelo activo y conduce al usuario a la localización exacta del error.
- Propiedades de los elementos (parte inferior derecha): permite cambiar las propiedades de modelos y elementos (importancias de tareas, cardinalidades, etc.)

3.3 Aplicando CSRML Tool: Jigsaw, una Actividad E-Learning Cooperativa

En esta sección se mostrará el uso de la aplicación CSRML Tool mediante el modelado de una actividad de *jigsaw* (puzzle) [27]. Esta es una técnica de aprendizaje cooperativo en la que los estudiantes realizan algún estudio a nivel individual sobre un problema propuesto, tras lo cual, enseñarán al resto de los integrantes de su grupo lo que han aprendido, compartiendo cada uno su punto de vista individual sobre el problema. Antes de realizarse esta actividad, se establecen los grupos (de entre 4 y 6 alumnos) y a cada uno de los alumnos se les proporciona una "pieza del puzzle" sobre la que aprender o investigar. Por ejemplo, una tarea de lectura considerablemente larga podría ser dividida en seis partes más pequeñas, convirtiéndose cada alumno en el *experto* de su parte asignada. Finalmente, las "piezas del puzzle" se juntarán cuando el grupo vuelva a reunirse y los alumnos compartirán su conocimiento.

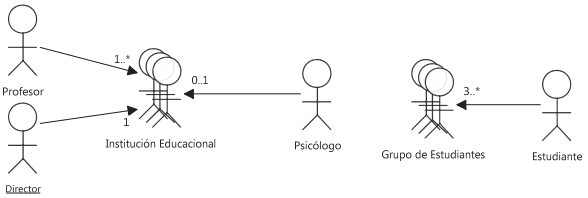


Fig. 8. Diagrama de jerarquía de grupo (GHD)

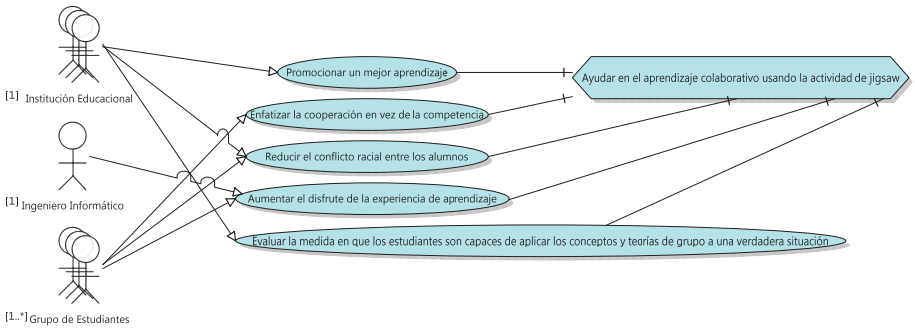


Fig. 9. Diagrama de objetivos del sistema (SGD)

Siguiendo la secuencia de diagramas mostrada en la Fig. 3, en primer lugar debe definirse el GHD (Fig. 8) para identificar los actores y actoresgrupo, así como para asignar actores a sus correspondientes grupos por medio de los enlaces de participación. Como puede observarse, una *Institución Educacional* consta de un

Director, uno o más *Profesores* y opcionalmente, un *Psicólogo*. Además, este grupo tiene un líder (*Director*, subrayado en el diagrama), pero no el *Grupo de Estudiantes*.

Después del GHD, debemos definir el SGD (Fig. 9) para especificar los objetivos del sistema. Estos objetivos son logrados mediante la tarea principal del sistema *Ayudar en el aprendizaje colaborativo usando la actividad de jigsaw* y la participación de varios actores (o grupo de actores). Por ejemplo, el objetivo *Aumentar el disfrute de la experiencia de aprendizaje* será logrado gracias a la participación de los *Grupos de Estudiante* y del *Ingeniero Informático*, quien podría añadir características multimedia a la interfaz con este fin. Como puede observarse en la cardinalidad establecida entre corchetes, uno o más grupos de estudiantes participarán en el logro de este objetivo, pero sólo participará un ingeniero informático.

Una vez definidos el GHD y el SGD, la tarea principal del sistema debe ser refinada por medio del RD (Fig. 10). En este diagrama se descompone la tarea principal en sub-tareas y softgoals de calidad. Además, se asigna las responsabilidades por medio de los enlaces de responsabilidad. Por ejemplo, el rol *Gestor del jigsaw* (representado por un *Profesor* mientras la actividad de jigsaw está realizándose) es responsable tanto de la tarea principal, como de *Crear Grupos*.

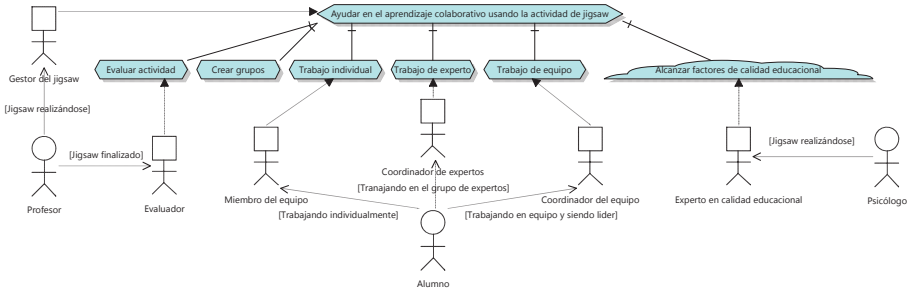


Fig. 10. Diagrama de responsabilidades (RD)

Cada una de las cinco sub-tareas (véase Fig. 10) en las que se descompone la tarea principal se refinará en un TRD diferente (debido las limitaciones de espacio mostraremos solamente uno de ellos). Por ejemplo, en la Fig. 11 se muestra cómo la tarea abstracta *Trabajo de equipo* se descompone en nuevos objetivos y sub-tareas (tanto individuales como 3C). Aquí, la involucración de roles (interpretados por actores) en tareas se realiza por medio de enlaces de participación. En este caso, para realizar la tarea *Hacer informe*, se necesitan dos o más *Miembros del equipo*. Como puede observarse, este enlace de participación está relacionado con un recurso de awareness debido a que para realizar dicho informe, los alumnos deben ser conscientes de la actividad de los otros miembros de su equipo (el comportamiento típico de un editor de textos colaborativo). Analizando el recurso de awareness de manera expandida se puede comprobar que para la realización de esta tarea, los usuarios tienen que ser conscientes de qué están haciendo los otros usuarios y con qué artefacto (párrafo, imagen, tabla, etc.) están trabajando (*What-Action* y *What-Artifact*). Además, los usuarios deberían también percibir con quién están colaborando y dónde están trabajando (*Who-Identity* y *Where-Location*).

Adicionalmente, es deseable ser consciente de la intención del resto de usuarios (*What-Intention*), y sería interesante tener información de lo que están viendo exactamente (*Where-View*) y de los elementos que tienen a su alcance (*View-Reach*). Además, en este TRD podemos observar cómo se han especificado las distintas importancias entre tareas, teniendo por ejemplo *Enviar informe* una importancia normal y *Hacer informe* una importancia muy alta.

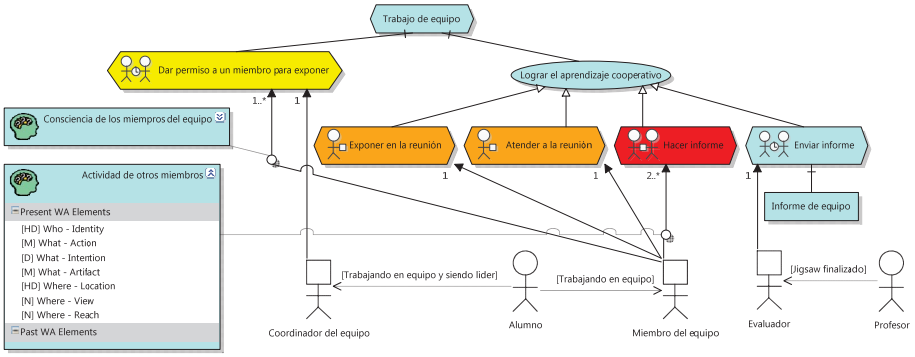


Fig. 11. Diagrama de refinamiento de tareas (TRD)

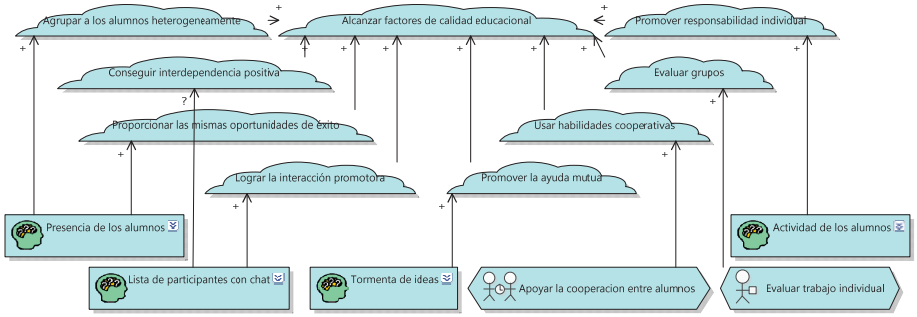


Fig. 12. Diagrama de factores de calidad (QFD)

Finalmente, la Fig. 12 muestra el último de los diagramas propuestos en CSRML, el QFD. En este modelo se muestran los factores de calidad que contribuyen a realizar la actividad de *jigsaw* con un nivel de calidad elevado. Estos factores se representan como softgoals y se relacionan con el factor de calidad principal mediante contribuciones. El logro de estos softgoals de calidad se alcanzará de diferentes formas. Por ejemplo, el softgoal *Agrupar a los alumnos heterogéneamente* será logrado por medio del recurso de awareness *Presencia de los alumnos*, que podría ser implementado con un *video embodiment widget*.

4 Conclusiones y trabajo futuro

En trabajos anteriores, basándonos en un proceso guiado por evaluaciones empíricas se definió el lenguaje CSRML [8, 21] como una extensión del lenguaje Goal-Oriented *i** [9, 10] que permitirá la especificación de los requisitos de un sistema colaborativo,

que son difíciles con las técnicas clásicas de IR [6, 7]. Esto es debido a la necesidad especial de percepción (*Workspace Awareness*) que requieren los usuarios de un sistema colaborativo acerca de, por ejemplo, quién está disponible para colaborar, qué están haciendo el resto de usuarios, quién y cuándo se realizó determinada acción.

En este trabajo se presenta CSRML Tool, la herramienta que permite la especificación de requisitos utilizando el lenguaje CSRML que ha sido creado mediante la implementación del meta-modelo de CSRML en Visualization and Modeling SDK (VMSDK). CSRML Tool permite la especificación de los requisitos de un sistema colaborativo por medio de los cinco tipos de diagramas que forman CSRML basada en una interfaz de usuario sencilla y totalmente integrada con Windows y Visual Studio. Esta herramienta permite validar los modelos creados, evitando errores e incoherencias y facilitando la futura modificación de los mismos.

Nuestro trabajo futuro consiste en la creación de una nueva versión de CSRML que permita la creación de distintas vistas sobre un mismo modelo, ya que actualmente VMSDK carece de esa funcionalidad de forma nativa. Además, otra característica que será añadida a la herramienta será la trazabilidad con técnicas de desarrollo de interfaces de usuario para sistemas colaborativos, como CIAM [25], obteniéndose así una metodología completa para el desarrollo de este tipo de sistemas. Finalmente, también forma parte de nuestro trabajo futuro la realización de experimentos con usuarios a fin de validar su usabilidad.

Agradecimientos. Este trabajo ha sido parciamente financiado por el proyecto PEII09-0054-9581 de la Junta de Comunidades de Castilla-La Mancha y por el proyecto DESACO (TIN2008-06596-C02-01) del Ministerio de Ciencia e Innovación.

Bibliografía

1. Google Inc.: Google DoubleClick Ad Planner 1000 most-visited sites on the web, http://www.google.com/adplanner/static/top1000/index.html.
2. Google Inc.: Google Docs, https://docs.google.com.
3. Schmidt, K., Bannon, L.: Taking CSCW seriously. Computer Supported Cooperative Work. 1, 7-40 (1992).
4. Ellis, C.A., Gibbs, S.J., Rein, G.: Groupware: some issues and experiences. Communications of the ACM. 34, 39-58 (1991).
5. Gutwin, C., Greenberg, S.: A Descriptive Framework of Workspace Awareness for Real-Time Groupware. Computer Supported Cooperative Work. 11, 411-446 (2002).
6. Teruel, M.A., Navarro, E., Lopez-Jaquero, V., Montero, F., Gonzalez, P.: An empirical evaluation of requirement engineering techniques for collaborative systems. 15th Annual Conference on Evaluation & Assessment in Software Engineering (EASE'11). pp. 114-123. IET, Durham, UK (2011).
7. Teruel, M.A., Navarro, E., López-Jaquero, V., Montero, F., González, P.: A Comparative of Goal-Oriented Approaches to Modelling Requirements for Collaborative Systems. 6th International Conference on Evaluation of Novel Software Approaches to Software Engineering (ENASE'11). pp. 131-142. SciTePress, Beijing, China (2011).
8. Teruel, M.A., Navarro, E., López-Jaquero, V., Montero, F., González, P.: CSRML: A Goal-Oriented Approach to Model Requirements for Collaborative Systems. 30th

International Conference on Conceptual Modeling (ER'11). pp. 33-46. Springer Berlin Heidelberg, Brussels, Belgium (2011).
9. Castro, J., Kolp, M., Mylopoulos, J.: A requirements-driven development methodology. 13th Int. Conf. On Advanced Information Systems Engineering (CAiSE'01). pp. 108-123. Springer-Verlag, London, UK (2001).
10. López, L., Franch, X., Marco, J.: Making Explicit Some Implicit i* Language Decisions. 30th International Conference on Conceptual Modeling (ER'11). pp. 62-77 (2011).
11. Ernst, N., Yu, Y., Mylopoulos, J.: Visualizing Non-Functional Requirements. 2006 First International Workshop on Requirements Engineering Visualization (REV'06 - RE'06 Workshop). pp. 2-2. IEEE (2006).
12. Gronback, R.C.: Eclipse Modelling Project: A Domain-Specific Languaje Toolkit. Addison-Wesley (2009).
13. Malta, Á., Soares, M., Santos, E., Paes, J., Alencar, F., Castro, J.: iStarTool: Modeling requirements using the i* framework. CEUR Proceedings of the 5th International i* Workshop (iStar 2011). pp. 163-165 (2011).
14. Kolp, M., Faulkner, S., Wautelet, Y., Nguyen, T., Coyette, A., Achbany, Y., Hoang, H., Do, T.: DesCARTES Architect. Business Driven Software Design. Forum des Recherches Doctorales. , Institut d'administration et de gestion (IAG) (2005).
15. Grau, G., Franch, X., Avila, S.: J-PRiM: A Java Tool for a Process Reengineering i* Methodology. 14th IEEE International Requirements Engineering Conference (RE'06). pp. 359-360. IEEE (2006).
16. van Lamsweerde, A.: Goal-oriented requirements engineering: a guided tour. Fifth IEEE International Symposium on Requirements Engineering (RE'01). pp. 249-262. IEEE Comput. Soc, Washington DC, USA (2001).
17. Cockburn, A.: Writing Effective Use Cases. Addison-Wesley Professional (2000).
18. Booch, G., Rumbaugh, J., Jacobson, I.: The Unified Modeling Language User Guide. Addison-Wesley Professional (2005).
19. Finkelstein, A., Kramer, J., Nuseibeh, B., Finkelstein, L., Goedicke, M.: Viewpoints: A Framework for Integrating Multiple Perspectives in System Development. International Journal of Software Engineering and Knowledge Engineering. 2, 31-57 (1992).
20. Cysneiros, L.M., Yu, E.: Non-Functional Requirements Elicitation. In: do Prado Leite, J.C.S. and Doorn, J.H. (eds.) Perspectives on Software Requirements. Kluwer (2003).
21. Teruel, M.A., Navarro, E., López-Jaquero, V., Montero, F., González, P.: Modelado de Requisitos de Sistemas Colaborativos con CSRML. XVI Jornadas de Ingeniería del Software y Bases de Datos (JISBD'11). pp. 639-652. , A Coruña, Spain (2011).
22. Teruel, M.A., Navarro, E., López-Jaquero, V., Montero, F., González, P.: Assessing the Understandability of Collaborative Systems Requirements Notations: an Empirical Study. 1st Int. Work. on Empirical Requirements Engineering (EmpiRE'11). pp. 85-92, Trento, Italy (2011).
23. Teruel, M.A., Navarro, E., López-Jaquero, V., Montero, F., Jaen, J., González, P.: Analyzing the Understandability of Requirements Engineering Languages for CSCW Systems: A Family of Experiments. Information and Software Technology. (2012).
24. Özgür, T.: Comparison of Microsoft DSL Tools and Eclipse Modeling Frameworks for Domain-Specific Modeling In the context of the Model-Driven Development, (2007).
25. Molina, A.I., Redondo, M.A., Ortega, M., Hoppe, U.: CIAM: A methodology for the development of groupware user interfaces. JUCS. 14, 1435–1446 (2008).
26. Aitken, P.G.: .Net Graphics and Printing. Optimax Pub (2003).
27. Pozzi, F.: Using Jigsaw and Case Study for supporting online collaborative learning. Computers & Education. 55, 67-75 (2010).

Una arquitectura para el desarrollo de videojuegos educativos con actividades colaborativas

N. Padilla Zea, P. Paderewski, F.L. Gutiérrez Vela, N. Medina Medina

Departamento de Lenguajes y Sistemas Informáticos. Laboratorio de Investigación en Videojuegos y E-learning. Grupo GEDES. Universidad de Granada.
{npadilla,patricia,fgutierr,nmedina}@ugr.es

Abstract. En este trabajo se abordan dos de los elementos claves de la técnica de aprendizaje colaborativo y de su utilización como base del proceso de aprendizaje en un videojuego educativo. El primero se refiere a la necesidad de los profesores de tener un registro de la evolución de sus estudiantes, no sólo del resultado final, sino también del proceso seguido en las tareas afrontadas durante el aprendizaje. El segundo objetivo, muy ligado al anterior, se refiere a la necesidad de analizar la interacción que tiene lugar entre los jugadores con el fin de evaluar cómo ésta afecta al aprendizaje. Para ello, se presenta una arquitectura que utiliza un conjunto de modelos que permite desarrollar videojuegos educativos efectivos, y concretamente, videojuegos educativos en grupo que implementen tareas colaborativas. Esta arquitectura, llamada PLAGER-VG (*PLAtform for managinG Educational multiplayeR Video Games*), permite diseñar, monitorizar y adaptar procesos de aprendizaje colaborativo soportados por videojuegos.

Keywords. Videojuegos educativos, Aprendizaje Colaborativo, Arquitecturas, Modelado.

1 Introducción

El juego, tanto en niños como en adultos, se ha estado utilizando desde hace tiempo como medio de aprendizaje y muchos expertos del ámbito educativo han defendido y presentado los numerosos beneficios que aporta este uso [1][2]. Actualmente, debido a la evolución que ha sufrido la tecnología y a cómo ésta es usada sobre todo por los menores, los videojuegos son una forma especial de juego que puede ayudar a mejorar el proceso de aprendizaje de los estudiantes. Aunque es cierto que hay voces a favor y en contra del uso de estos sistemas, el número de investigaciones que apoyan el uso de videojuegos como herramienta de aprendizaje supera a los que defienden lo contrario. De hecho, existen numerosos estudios con resultados realmente satisfactorios, tanto utilizando videojuegos comerciales como aquellos que se diseñan específicamente para enseñar (juegos serios) [3].

En anteriores trabajos [4], hemos presentado el paradigma VGSCL (Aprendizaje colaborativo mediado por videojuegos, del inglés, *Video Game – Supported Collaborative Learning*) como una teoría basada en el CSCL (*Computer - Supported*

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 145-158, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

Collaborative Learning) que utiliza las numerosas bondades de los juegos como herramienta motivadora y mediadora del aprendizaje. Este paradigma pretende aunar los beneficios derivados de tres técnicas educativas suficientemente contrastadas (aprendizaje basado en juegos, aplicación de nuevas tecnologías y aprendizaje colaborativo), con el fin de obtener un proceso de aprendizaje lo más eficiente y satisfactorio posible, tanto desde el punto de vista del estudiante como del profesor.

La unificación de estas tres técnicas en una sola hace que el diseño y el análisis de este proceso de aprendizaje sean mucho más complejos. Para que el sistema resultante sea suficientemente efectivo, es necesario que se diseñen todos los elementos adecuadamente y que se introduzca un mecanismo de monitorización y análisis capaz de identificar todos los elementos y procesos relevantes, así como de obtener una traza de ellos y combinarlos para obtener información relevante. Por este motivo, cuestiones como observar la evolución de los estudiantes a lo largo del tiempo, permitir que se introduzcan cambios que mejoren el proceso y medir y analizar la colaboración que está teniendo lugar en el grupo son fundamentales. Para lograrlo, es necesario registrar y organizar toda la información que pueda ser importante para la posterior evaluación. La única forma de conseguirlo es mediante una adecuada y exhaustiva modelización del sistema que permita almacenar todos los datos de interés acerca de cada jugador, de los grupos que colaboran y del proceso de juego / aprendizaje. Así, es necesario un sistema que permita que se realicen las adaptaciones necesarias en el proceso de aprendizaje con el fin de mejorar los resultados obtenidos por los estudiantes.

Por otra parte, en el contexto educativo en el que se utilizan los videojuegos educativos, se hace evidente la necesidad de un marco de trabajo bien definido que permita diseñar videojuegos educativos efectivos, y en concreto, videojuegos educativos en grupo que favorezcan la colaboración. Este marco de trabajo debe permitirnos diseñar, monitorizar y adaptar procesos de aprendizaje colaborativo soportados por videojuegos. En este artículo, presentamos una propuesta de arquitectura para conseguir dicho marco de trabajo.

El trabajo que presentamos a continuación se organiza del modo siguiente. En primer lugar, analizamos los trabajos relacionados tanto con el proceso de diseño como con arquitecturas de soporte de videojuegos educativos, obteniendo un conjunto de requisitos que deben tener ambos. Continuamos presentando los modelos y la arquitectura basada en éstos que forman parte de nuestra propuesta, así como una evaluación de la misma. Terminaremos con las conclusiones y trabajos futuros.

2 Trabajos Relacionados

El Aprendizaje Colaborativo Soportado por Computador (CSCL) ha sido ampliamente estudiado desde diversos campos de investigación, principalmente desde el campo de la psicología y de la informática. Las distintas investigaciones realizadas han proporcionado resultados muy positivos que nos llevan a concluir que la aplicación de las técnicas asociadas a esta disciplina aporta beneficios adicionales al proceso de aprendizaje. Sin embargo, surge el problema de comprobar que los estudiantes que están trabajando en grupo están realmente colaborando. Esto nos lleva

a tener que realizar un análisis del trabajo que se produce en el grupo, es decir, nos interesa saber si el grupo de estudiantes está colaborando o no y en qué medida.

Igualmente importante nos parece la utilización de los videojuegos como herramienta de aprendizaje (Game Based Learning). Aproximaciones como los videojuegos educativos o los juegos serios se han utilizado con éxito para incentivar la motivación de los estudiantes dentro de sus procesos de aprendizaje.

Un inconveniente que encontramos en los videojuegos educativos es la falta de equilibro entre los contenidos educativos y los aspectos lúdicos que se proponen. Una de las razones por las que se produce esta situación es que el proceso de diseño y el sistema de soporte para desarrollar dichos videojuegos no están adecuadamente definidos, lo que implica una dificultad adicional tanto para profesores como para desarrolladores. Con el fin de estudiar esta problemática, hemos hecho una revisión de distintas propuestas acerca del diseño de videojuegos educativos, así como de las arquitecturas que se proponen para su implementación. Como veremos, no existen demasiadas propuestas en este sentido y las que encontramos son específicas para problemas concretos, limitando el tipo de juego que se puede diseñar.

2.1 Trabajos Relacionados con el Proceso de Diseño de Juegos Educativos

La metodología propuesta en EMERGO [5] guía el desarrollo de juegos serios basados en escenarios. Los autores definen este tipo de juegos como un entorno simulado de tareas modeladas sobre situaciones de la vida real, que a menudo incluye una secuencia de aprendizaje que supone procesos complejos de toma de decisiones, estrategias de resolución de problemas, razonamiento inteligente y otras habilidades cognitivas complejas.

EDoS [6] es un entorno interactivo de diseño de juegos serios para la enseñanza de competencias relacionadas con la ingeniería, que proporciona al equipo de desarrollo un método para crear juegos de forma eficiente, visual y estructurada. Permite la colaboración entre docentes y técnicos, de tal forma que los conocimientos de ambas especialidades se combinen de forma adecuada.

En el congreso ECGBL 2010 (*European Conference on Game – Based Learning*) se presentó un proceso de diseño en seis pasos para juegos serios [7] que no tienen que seguirse en un orden estricto. Está pensado para desarrollar juegos serios que tienen como objetivo enseñar competencias profesionales, por ello el primer paso es la especificación de objetivos pedagógicos: se extrae y formaliza el conocimiento específico del dominio de conocimiento que se desea practicar. A continuación, se elije un modelo predefinido de juego (segundo paso) y se crea el escenario pedagógico (tercer paso). El experto pedagógico y el diseñador de juegos trabajan juntos, ya que tendrán que unificarlo en un escenario de diversión (cuarto paso). El quinto paso es realizar una simulación con el objetivo de evaluar el juego en términos de calidad pedagógica. Finalmente, el sexto paso consiste en ilustrar cada escena con todos los detalles e interacciones que se incluirán en el juego.

Sauvé [8] presenta una propuesta filosófica para diseñar videojuegos educativos online de forma sencilla, que permite crear diferentes juegos partiendo de la misma estructura de juego. Las herramientas que se ponen a disposición de los profesores permiten fijar los valores de los parámetros, generar las reglas para definir los movimientos de los jugadores, crear el material educativo, definir los criterios para

fijar el fin del juego y los ganadores, y elaborar las herramientas para la revisión y evaluación del juego.

De entre las propuestas que hemos valorado, encontramos que prácticamente todas involucran en el proceso de diseño tanto a profesores como a diseñadores de juegos con el fin de poder diseñar de forma adecuada tanto la parte educativa como la de juego. Además, algunas de las propuestas especifican distintos roles más específicos, que hacen referencia a cada uno de los componentes que es necesario definir, sobre todo en la parte de juego (diseñadores gráficos, diseñadores de contenidos, etc.). Muchas de las propuestas introducen mecanismos de reutilización de los distintos componentes ya definidos para minimizar el coste en el diseño y construcción de este tipo de sistemas. También coinciden en que se debe comenzar el proceso de diseño con la especificación del contenido educativo, ya que el objetivo principal es el aprendizaje y después, se componener la historia del juego más adecuada a los conocimientos que se quieren practicar.

Todas estas características son adecuadas en la especificación de un método de diseño, pero también hemos encontrado en las propuestas estudiadas algunos aspectos que pensamos que son mejorables. Uno de ellos es que todas las propuestas están pensadas para juegos serios, cuyos objetivos difieren en algunos aspectos de los de los videojuegos educativos [9]. Por esto, en los juegos serios es normal encontrar procesos basados en mecanismos de simulación de comportamientos, reacciones, procesos, etc. donde la conjunción de la componente educativa y la componente lúdica suele ser bastante similar en todos los casos. Esto provoca que la generalización de los procesos a videojuegos educativos sea compleja y, por tanto, se reduzcan los beneficios que se podrían obtener de un proceso de diseño estructurado.

Por otra parte, pensamos que existe una dependencia demasiado estricta del mecanismo de definición del contenido lúdico con el educativo en la mayoría de las propuestas. Las cuestiones relativas a los niveles de jugabilidad [10] alcanzados durante el juego, deben tenerse en cuenta desde las etapas más tempranas del desarrollo con el fin de conservar las características beneficiosas de los videojuegos.

Por último y no menos importante, consideramos que es necesario definir de forma concreta la información necesaria en cada uno de los pasos que componen el proceso de diseño. Así, es más fácil guiar tanto a profesores como a diseñadores para que se llegue a una definición del juego sin ambigüedades respecto a los objetivos (lúdicos y educativos) que se desean conseguir y cómo deben conseguirse. En las propuestas analizadas no se especifica claramente la información concreta que requiere cada uno de los pasos.

2.2 Trabajos Relacionados con la Arquitectura de Soporte para Juegos Educativos

En la literatura encontramos un conjunto de arquitecturas que diversos autores han propuesto como mecanismo de soporte al diseño y ejecución de videojuegos educativos. En la tabla 1 podemos encontrar los resultados de un análisis que se ha realizado sobre las propuestas más destacadas, donde × significa que la arquitectura carece de dicho aspecto y – indica que no es aplicable. En concreto, los aspectos que más nos han interesado se centran en si permite desarrollar juegos de distinto género como e-Adventure [11][12] o AEINS [13], si tiene en cuenta la jugabilidad, si se tiene en cuenta a los profesores en su desarrollo, qué tipo de modelo educativo utiliza

(implícito o explícito), si permite que se realice a posteriori un análisis del aprendizaje, si tienen en cuenta el aprendizaje colaborativo y permite realizar un análisis de la colaboración realizada por los jugadores, si implementa la competencia entre grupos y si permite que se realicen cambios, es decir, si genera juegos adaptables.

Tabla 1 Comparación de distintas arquitecturas

	NUCLEO [14]	**e-Adventure** [11][12]	**EGA** [15]	**AEINS** [13]	**PLAGER-VG**
Distintos géneros de juegos	×	☺	☺	×	☺
Evalúa la jugabilidad	×	×	×	×	☺
Involucra a profesores	☺	☺	×	×	☺
Tipo de modelo educativo	Implícito	Implícito	Explícito	Implícito	Explícito
Análisis del aprendizaje	×	☺	×	☺	☺
Colaboración	☺	×	×	×	☺
Análisis de la colaboración	×	-	-	-	☺
Competición entre grupos	☺	-	-	-	☺
Adaptación	☺	☺	×	☺	☺

La mayoría de las arquitecturas evaluadas presenta un conjunto de deficiencias que dificultan su incorporación en los procesos reales de aprendizaje que tienen lugar en las aulas, como son:

- No tienen en cuenta la estructuración habitual de los contenidos que se produce en los procesos de aprendizaje.
- No tienen suficientemente en cuenta la figura del profesor en el proceso de planificación de contenidos y no ofrecen ninguna herramienta que permita hacer cambios en el proceso estándar.
- No existe un modelado de actividades que permita identificar claramente la correspondencia entre las actividades educativas y las lúdicas.
- Es difícil conseguir una traza detallada de lo que aprende el estudiante en cada parte del juego debido a que existe una falta de correspondencia entre lo pedagógico y lo lúdico.
- La mayoría de ellas, salvo la EGA, no considera aspectos de jugabilidad en el proceso de juego, cuestión que a la postre revertirá, en numerosas ocasiones, en la obtención de juegos poco jugables o en la necesidad de rediseñar dichos juegos.

La propuesta más cercana a nuestro trabajo ‹e-Adventure› [16][17], ya que es la única propuesta en nuestra revisión bibliográfica que integra un proceso de diseño y una arquitectura de soporte para el diseño de videojuegos educativos y, además, está enfocada al aprendizaje en grupo. ⟨e-Adventure⟩ es un marco de trabajo para el desarrollo de aventuras gráficas (*point and clic*) centrado principalmente en el

desarrollo de herramientas educativas, aunque también se puede usar para videojuegos no educativos. Para el proceso de diseño presentan un conjunto de guías de estilo generales que facilita la integración de juegos adaptativos y medibles en entornos de aprendizaje online. La arquitectura [11] que le da soporte se comporta como un middleware de dos capas que conecta el LMS (Learning Management System) con los juegos. La primera capa es la Capa de Comunicación (CL, Communication Layer), que es responsable de establecer y gestionar el canal de comunicación entre el LMS y el juego. Por tanto, la CL se encarga de establecer la comunicación, enviar y recibir datos desde y hacia el VLE (Virtual Learning Environment) y desconectar. Esta capa necesita entender distintas especificaciones para permitir la interconexión con distintos VLE. La segunda capa, llamada Capa de Adaptación del Juego (GAL, Game Adaption Layer), monitoriza la interacción entre el juego y el estudiante y recopila esta información para mantener un registro de la actividad del estudiante. Además, la GAL usa los servicios proporcionados por la CL para obtener información acerca del estudiante, el curso, etc. Esta información se utiliza para analizar la experiencia de juego de forma transparente al usuario y puede utilizarse también para evaluar la actuación del estudiante.

Sin embargo, en esta propuesta encontramos confuso el modo de estructurar el contenido educativo y la forma de relacionar cada contenido educativo con los componentes lúdicos que se presentan en el juego. Es decir, se hace más énfasis en la parte de juego. Además, tampoco existe una relación clara entre las guías de diseño y la arquitectura de soporte. Por otra parte, aunque se realiza una monitorización de los jugadores, esta monitorización está limitada por las posibilidades que ofrece el LMS utilizado y, además, no es capaz de informar acerca de cada elemento de conocimiento en relación a la actividad del estudiante en el juego.

3 Requisitos para una Propuesta Integrada

Vistas las diferentes propuestas analizadas en el punto anterior, hemos identificado un conjunto de requisitos que pensamos que debe reunir un proceso de diseño para videojuegos educativos con actividades colaborativas y la arquitectura que se utilice como soporte. Esto nos ha servido para poder realizar una propuesta integrada.

Estos requisitos, para el proceso de diseño, son:

- Cubrir todo el proceso de diseño del juego, teniendo en cuenta los aspectos educativos, los lúdicos y las relaciones que existan entre ellos.
- Definir el conjunto de elementos básico para cada uno de estos aspectos, así como la información necesaria para definirlos completamente.
- Especificar el dominio válido de cada uno de los atributos de los elementos definidos en el sistema.
- Incluir un proceso sistemático y claro de definición de cada uno de estos elementos, determinando qué fases se deben abordar, en qué orden, qué información se debe completar en cada una de dichas fases y qué documentos se generan para cada una de estas fases.
- Establecer claramente las relaciones que existen entre los distintos elementos y facilitar al usuario la definición de dichas relaciones.

Y para la arquitectura que dé soporte a este tipo de juegos, los requisitos son:

- Cubrir las distintas funcionalidades requeridas en un proceso de aprendizaje mediado por videojuegos educativos, que al menos pasan por: definir los contenidos educativos, definir los contenidos lúdicos y monitorizar el aprendizaje.
- Ofrecer servicios que faciliten la definición del contenido educativo por parte de los profesores ya que, muchas veces, no están familiarizados con el uso de nuevas tecnologías.
- Ofrecer servicios que permitan la definición de los elementos del videojuego y de su relación con el contenido educativo, si existe.
- Permitir la personalización o adaptación del proceso de aprendizaje y tener en cuenta estas reglas en el proceso de juego asociado a dicho aprendizaje.
- Dar soporte a la ejecución del juego en sí mismo y permitir la monitorización de la actuación de los jugadores durante el juego.

Además, interesa que el proceso de diseño y la arquitectura de soporte estén claramente relacionados, de tal forma que los desarrolladores conozcan los elementos de la arquitectura que se ven afectados en cada parte del diseño. Esto permite mantener la coherencia tanto del proceso de diseño como de la arquitectura cuando se introduzcan cambios en alguna parte de ellos.

4 Desarrollo bajo VGSCL

El diseño de sistemas usando modelos permite realizar abstracciones explícitas del sistema independientes de la implementación posterior, lo cual aporta un nivel adecuado de flexibilidad que permite el posterior mantenimiento y reutilización del sistema y de la información almacenada en dichos modelos [18]. Nosotros hemos usado estas ideas para crear un proceso de diseño de videojuegos usando un para ello un conjunto de modelos. El proceso definido bajo el paradigma VGSCL (Video Games - Supported Collaborative Learning) permite modelar sistemas de aprendizaje colaborativo soportado por videojuegos [19]. El modelado de este tipo de sistemas tiene tres objetivos principales: 1) Involucrar al profesor dentro del proceso de diseño de los videojuegos, 2) enlazar los elementos propios del videojuego con los contenidos educativos y 3) facilitar la monitorización del proceso de aprendizaje que están desarrollando los estudiantes con objeto de estudiar dicho aprendizaje y permitir, si fuese necesario, adaptaciones en el proceso de juego que repercutan en una mejora de dicho aprendizaje. Para satisfacer este último objetivo es necesario modelar al estudiante desde tres perspectivas: aprendizaje, juego e interacción con otros compañeros. De esta forma, es posible obtener un conjunto de conclusiones a partir de la información recogida en los modelos y el análisis exhaustivo de dicha información. Por tanto, para implementar estos modelos es necesario considerar tres elementos: 1) los objetivos educativos y del videojuego; 2) las tareas educativas y del videojuego; y 3) la interacción entre los estudiantes / jugadores.

Los modelos implicados en el proceso de diseño, se puede dividir en los siguientes cuatro grupos.

4.1 Modelos que nos permiten definir y monitorizar el proceso educativo

En la parte educativa, los principales responsables en la definición de contenidos son los profesores o el equipo pedagógico. En los modelos incluidos en este grupo se definen las asignaturas que se van a impartir y la representación de los contenidos asociados a dichas asignaturas, en términos de objetivos y tareas que deben estar bien relacionadas de tal forma que se sepa qué tareas se han de completar para conseguir un determinado objetivos.

Definimos cuatro modelos distintos: 1) *Modelo de Área de Conocimiento*, para definir las áreas de conocimiento donde se encuadra el contenido educativo, 2) *Objetivo Educativo*, para definir cada uno de los objetivos que el estudiante debe superar; 3) *Modelo de Tarea Educativa*, para describir el conjunto de tareas incluidas en el proceso de aprendizaje; y 4) *Modelo educativo*, que instancia los objetivos y tareas educativos de acuerdo a una estrategia de aprendizaje concreta.

4.2 Modelos para especificar y monitorizar el contenido lúdico

Su objetivo es especificar el conjunto de actividades y objetivos lúdicos del videojuego que se va a utilizar para enseñar los contenidos educativos. Además, incluirán información para el profesor con el fin de permitir la monitorización de los procesos de juego y aprendizaje. Los conceptos que se manejan son paralelos a los que se han usado en los modelos del contenido educativo: los objetivos educativos son retos del videojuego y las tareas y actividades educativas son fases y niveles del videojuego.

Se definen los siguientes modelos: 1) *Modelo de Juego*, 2) *Modelos de Reto del Videojuego* y 3) *Modelos de Fases y Niveles del Videojuego*.

Los *Retos del Videojuego* son hitos que los jugadores deben superar. Para superar un reto, el jugador debe completar un conjunto de *Fases* y *Niveles*, a través de los cuales, además de superar el reto propuesto en el videojuego, el jugador va obteniendo distintos conocimientos de su currículum educativo.

4.3 Modelo para relacionar el contenido educativo y el lúdico

El *Modelo General de Objetivos y Tareas* define cómo los contenidos lúdicos satisfacen los requisitos educativos. Esta correspondencia es muy importante ya que todo contenido educativo tiene que estar relacionado con uno o más objetivos lúdicos (la inversa no tiene porqué darse). Así, el hecho de completar una Fase o Nivel en el videojuego es equivalente a completar una Tarea o Actividad educativa. De forma similar, superar un Reto en el videojuego (R_V)es equivalente a aprender el contenido del Objetivo Educativo (G_e)asociado.

Este modelo tiene dos niveles: el *Nivel Educativo* - L_E (capa inferior, Figura 1) y Nivel de Videojuego – L_V (capa superior, Figura 1). De tal forma que una Fase o Nivel del Videojuego, Tv_B, implementa una Tarea o Fase Educativa, Te_A, si Tv_B en el videojuego es útil para enseñar la Tarea Educativa Te_A. Esta relación se representa por medio de una línea discontinua que conecta las tareas de ambos niveles. Por ejemplo, Tv_3 y Tv_6 implementan Te_4 en la Figura 1.

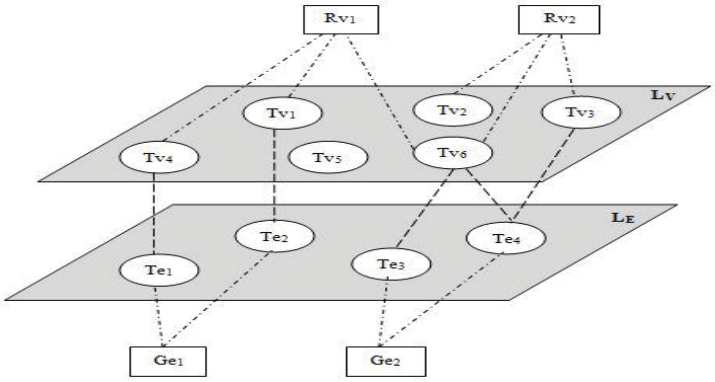


Figura 1 Modelo General de Objetivos y Tareas

4.4 Modelos de usuarios

Permiten monitorizar a los estudiantes mientras juegan, tanto de forma individual como en grupo. Para ello, tenemos tres modelos: *Modelo de Estudiante-Jugador, Modelo de Grupo* y *Modelo General de Grupo*. El primero está formado por cuatro perspectivas, cada una de ellas con información sobre los siguientes aspectos: personal, educativo, videojuego e interacción. La perspectiva personal contiene información general acerca del usuario. La perspectiva educativa está relacionada con los logros educativos del estudiante y contiene información acerca de los Objetivos y Tareas Educativas que cada estudiante debe realizar, está realizando, o ha finalizado. La perspectiva de videojuego tiene como objetivo adaptar el juego a las posibilidades del jugador para evitar situaciones en la que las dificultades que el estudiante tiene con el juego provoquen dificultades educativas. Y, la última, la perspectiva de interacción, está relacionada con la forma en la que los usuarios interactúan con el resto de compañeros del grupo.

El *Modelo de grupo*, también consta de cuatro perspectivas: la primera, la perspectiva de identificación, incluye meta-información acerca del grupo; la segunda, la perspectiva educativa, permite conocer el aprendizaje conseguido por el grupo; la tercera, la perspectiva de videojuego, incluye las preferencias del grupo respecto al juego; y, finalmente, la perspectiva de interacción, desde la que se analiza cómo el grupo interactúa como un todo.

Por último, el *Modelo General de Grupo*, define tipos de grupos con características específicas que luego se pueden instanciar para un conjunto concreto de estudiantes.

5 Arquitectura PLAGER-VG

PLAGER-VG es una arquitectura modular que, usando los modelos vistos en el apartado anterior, da soporte a procesos de aprendizaje mediados por videojuegos educativos con actividades colaborativas permitiendo el diseño, ejecución y monitorización de este tipo de aplicaciones. La parte de generación del juego podría

realizarse utilizando alguna propuesta de las existentes como la de [20 E.Montero y Carsí], donde se facilita esta generación independientemente de la plataforma. Para ello, proporciona cinco funciones principales:

- Da soporte al proceso de diseño integral definido para sistemas VGSCL.
- Facilita la adaptación y personalización de los procesos de aprendizaje colaborativos mediados por videojuegos.
- Da soporte al proceso de juego en sí mismo, ya que coordina su ejecución.
- Incorpora un mecanismo de monitorización y análisis, tanto del proceso de juego como del proceso de aprendizaje.
- Permite la gestión y monitorización de los grupos creados para las actividades de aprendizaje colaborativo.

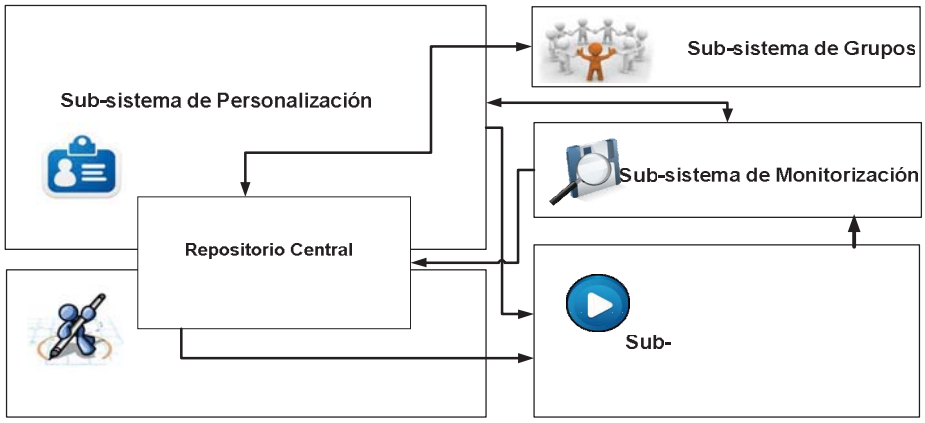


Figura 2 Arquitectura PLAGER-VG

Cada una de estas funciones principales constituye un sub-sistema independiente en la arquitectura (Figura 2), y se han denominado como sigue:

- *Sub-Sistema de Diseño*, para facilitar el diseño de los juegos educativos de los que se dispondrá para el proceso de aprendizaje.
- *Sub-Sistema de Personalización*, que facilita la adaptación y personalización del proceso de aprendizaje y de los juegos particulares para cada uno de los jugadores o grupos.
- *Sub-Sistema de Juego*, para controlar el juego y mantener las restricciones establecidas por el profesor para cada uno de los estudiantes o grupos.
- *Sub-Sistema de Monitorización*, que registra los eventos de interés, los procesa y envía la información recopilada a cada uno de los procesos encargados de su tratamiento.
- *Sub-Sistema de Grupos*, para diseñar los grupos necesarios e instanciarlos para cada juego.

Como se ve en la Figura 2, no son sub-sistemas aislados, sino en continua interacción. Así, tanto el contenido educativo como el juego se diseñan usando la funcionalidad incluida en el *Sub-sistema de Diseño*. Los componentes diseñados se almacenan en el *Repositorio Central*. El *Sub-sistema de Diseño*, por medio de este repositorio, se comunica con el *Sub-sistema de Personalización*, que accede a los elementos diseñados y los adapta a cada usuario del sistema. Estas modificaciones, que personalizan tanto el proceso de aprendizaje como el proceso de juego para cada

jugador, se plasman también en dicho repositorio. Para ello, el *Sub-sistema de Personalización* se comunica con el *Sub-Sistema de Juego* que, dado un conjunto de especificaciones educativas para un estudiante o grupo, genera una instancia de juego personal que se ejecutará de forma coherente con las restricciones educativas indicadas. Durante el juego, se producirá un conjunto de eventos de interés, tanto a nivel educativo como lúdico, que serán recogidos por el *Sub-sistema de Monitorización* para su procesamiento. Como resultado de este procesamiento, se generará un conjunto de recomendaciones que se comunican al *Sub-sistema de Personalización*, que se encargará de implementarñas. En el repositorio del sistema se realizarán los cambios referentes al diseño de actividades. Finalmente, el *Sub-sistema de Grupos* gestiona tanto el diseño como la creación de los grupos y almacena esta información que permite gestionar las actividades colaborativas incluidas en el juego.

En la Tabla 2 se muestra la relación entre los distintos sub-sistemas de la arquitectura propuesta y los modelos vistos en el punto anterior.

Tabla 2 Relación de cada sub-sistema con los modelos

Sub-sistema de Diseño	Modelo de Áreas de Conocimiento Modelo de Objetivos Educativos Modelo de Tareas y Actividades Educativas Modelo Educativo Modelo de Videojuego Modelo de Retos del Videojuego Modelo de Fases y Niveles del Videojuego Modelo General de Objetivos y Tareas Modelo de Estudiante-Jugador Modelo de Grupo Modelo General de Grupo
Sub-sistema de Grupos	Modelo de Estudiante-Jugador Modelo de Grupo Modelo General de Grupo
Sub-sistema de Personalización	Modelo de Estudiante-Jugador Modelo de Grupo Modelo General de Objetivos y Tareas
Sub-sistema de Monitorización	-
Sub-sistema de Juego	Modelo General de Objetivos y Tareas

6 Nutri-Galaxy

A modo de ejemplo, se presenta el diseño de un reto en el videojuego Nutri-Galaxy, videojuego que se ha implementado como parte de un proyecto fin de grado en la Universidad del Cauca (Colombia) [21]. Es un videojuego educativo con actividades colaborativas que se ha desarrollado para ser ejecutado en PLAGER-VG. Este juego, cuyos objetivos educativos están relacionados con el aprendizaje de la función de nutrición, se ha implementado para ser ejecutado en PC. Está destinado a niños de 11 a 12 años, que jugarán en un entorno 2D. Los jugadores se agruparán de cuatro en cuatro y en cada nivel deberán superar dos mini-juegos (retos), uno individual y otro

en grupo, para poder avanzar a la siguiente fase o nivel. En cada nivel se presentan pacientes con problemas nutricionales más complejos y difíciles de superar.

En la Tabla 3 se muestra el modelo que representa la actividad del juego donde los jugadores deben vencer a la enfermedad que sufre Odín, un campesino cuya alimentación está basada en la comida rápida y no hace ningún ejercicio físico. En la actividad planteada participan los 4 jugadores. Es importante definir un modelo para cada reto del juego, relacionándolo con el contenido educativo asociado (Tabla 4).

Tabla 3 Actividad del juego donde se derrota a la enfermedad diagnosticada

Atributo	Valor
Identificador	VS0023
Nombre General	Tratar Odín
Descripción	Los 4 integrantes del grupo deberán dar el tratamiento adecuado a Odín. Si los jugadores seleccionan los alimentos adecuados para tratar la obesidad, la enfermedad comenzará a asustarse y debilitarse hasta desaparecer; si no, la enfermedad continuará recordando a los jugadores que han fallado.
Categoría	Puzle
Jugadores	4
Tipo	Null
Longitud	Media
Dificultad	Normal
Control de usuario	Si
Dimensión cultural	Indiferente
Recursos	Ninguno

Tabla 4 Modelo de Objetivo Educativo que se trabaja en el reto del juego VS0023

Atributo	Valor
Identificador	ET0004
Nombre General	Escoger alimentos
Área de conocimiento	Conocimiento del medio
Contenido educativo	Escoger qué alimentos se deben consumir según su composición nutricional

Figura 3 Pantalla del juego correspondiente a la tarea educativa ET0004

A partir de la definición del contenido educativo realizada en la Tabla 4, se define una actividad lúdica que permite al alumno realizar un proceso de aprendizaje más atractivo. Una vez definido el reto por medio del correspondiente modelo (Tabla 3), los diseñadores pueden incluirlo en el juego, tal como se muestra en la Figura 3.

7 Conclusiones y Trabajos Futuros

En este trabajo hemos presentado, tras un estudio de los trabajos relacionados, una arquitectura basada en modelos para el diseño de videojuegos educativos. Se ha prestado un interés especial en poder describir juegos con una componente colaborativa, ya que este aspecto puede favorecer el proceso de aprendizaje de los estudiantes.

Gracias a la funcionalidad incorporada en la arquitectura y a su diseño estructural, conseguimos un conjunto de ventajas, como poder gestionar las diferentes funciones de forma independiente, incorporar aspectos de aprendizaje colaborativo al proceso educativo completo o a partes concretas del mismo, gestionar estudiantes y grupos facilitándole al profesor la asignación de tareas y el análisis conjunto de los grupos, y personalizar el aprendizaje por medio de la selección de los contenidos que se desean trabajar a través del juego.

Actualmente, estamos trabajando en el sub-sistema de monitorización. Aunque incorpora un conjunto de agentes para la recogida de información, pensamos que dicho sistema puede mejorarse, por lo que se está planteando la incorporación de agentes que actúen de intermediarios entre la recogida de información y el envío de alertas al profesor, así como en la implementación de las mejoras propuestas en el sistema, es decir, de las acciones de adaptación.

Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Ciencia e Innovación de España como parte del proyecto VIDECO (TIN2011-26928) y el Vicerrectorado de Política Científica e Investigación de la Universidad de Granada.

Referencias

1. Rosas, R., Nussbaum, M., López, X., Flores, P., Correa, M.: Más allá del Mortal Kombat: diseño de videojuegos educativos. En: V Congreso Iberoamericano de Informática educativa (2000).
2. Mooney, C.: Theories of Childhood: An Introduction to Dewey, Montessori, Erikson, Piaget & Vygotsky. Redleaf Press (2000).
3. Morales, E.: El uso de videojuegos como recurso de aprendizaje en educación primaria y Teoría de la Comunicación. Diálogos de la Comunicación 78 (2009).
4. N. Padilla Zea, J. L. González Sánchez, F. L. Gutiérrez, M. J. Cabrera, P. Paderewski: From CSCL to VGSCL: A new approximation to Collaborative Learning. I International Conference on Computer Supported Education (CSEDU), pp.23–26 (2009).

5. Nadolski, R. J., Hummel, H. G. J., van der Brink, H. J., Hoefakker, R. E., Slootmaker, A., Kurvers, H. J., Storm, J.: EMERGO: A methodology and toolkit for developing serious games in higher educatio. Simulation & Gaming 39(3), 338-352 (2008).
6. Tran, C. D., George, S., Marfisi-Schottman, I.: EDoS: An authoring environment for serious games. Design based on three models. 4th European Conference on Game-Based Learning, pp. 393-402 (2010).
7. Marfisi-Schottman, I., George, S., Tarpin-Bernard, F.: Tools and methods for efficiently designing serious games. 4th European Conference on Game-Based Learning, pp. 226-234, (2010).
8. Sauvé, L.: Design tools for online educational games: Concepts and application. Pan, Z., Cheok, D. A., Müller, W., El Rhalibi, A. (eds.) Transactions on Edutainment II. LNCS, vol. 5660, pp. 187-202. Springer Berlín, Heidelberg (2009).
9. Gros, B.: Certezas e interrogantes acerca del uso de los videojuegos para el aprendizaje. Comunicación 7(1), pp. 251-264 (2009).
10. González Sánchez, J. L.: Jugabilidad: Caracterización de la experiencia del jugador en videojuegos. Tesis doctoral, Universidad de Granada (2010).
11. De Blanco, A., Torrente, J., Moreno-Ger, P., Fernández-Manjón, B.: A general architecture for the integration of educational video games in standards-compliant virtual learning environments. 9th IEEE International Conference on Advanced Learning Techmologies (ICALT), pp. 53-55. IEEE Computer Society, California (2009).
12. De Blanco, A., Torrente, J., Moreno-Ger, P., Fernández-Manjón, B.: Integrating Adaptive Games in Student-Centered Virtual Learning Environments. International Journal of Distance Education Technologies, 8(3), pp. 1-15 (2010).
13. Hodhod, R., Cairns, P., Kudenko, D.: Innovative Integrated Architecture for Educational Games: Challenges and Merits. Transaction on Edutainment V, LNCS 6530, pp. 1-34 (2011)
14. Sancho, P.: NUCLEO: Un sistema para el aprendizaje virtual colaborativo escenificado a través del rol multi-juego. Tesis doctoral, Universidad Complutense de Madrid (2010).
15. Hu, W.: A common architecture for educational games. En: Zhang, X., Zhing, S., Pan, Z., Wong, K., Yun, R. (eds.) Edutainment 2010. LNCS, vol. 6249, pp. 405-416. Springer Verlag Berlín, Heidelberg (2010).
16. Moreno-Ger, P., Sancho Thomas, P., Martínez-Ortiz, I., Sierra, J. L., Fernández-Manjón, B.: Adaptive Units of Learning and Educational Videogames. Journal of Interactive Media in Education 2007/5 (2007).
17. Moreno-Ger, P., Fuentes Fernández, R., Sierra, J. L., Fernández-Manjón, B.: Model-checking for Adventure Videogames. Information and Software Technology 51(3), pp. 564-580 (2009).
18. Molina, M.: Modeling commercial knowledge to develop advanced agent-based marketplaces for e-commerce. En: Klusch, M., Zambonelli, F. (eds.) 5th International Workshop on Cooperative Information Agents. LNAI, vol. 2182, pp. 196-201. Springer-Verlag, London (2001).
19. N.Padilla Zea. Metodología para el diseño de videojuegos educativos sobre una arquitectura para el análisis del aprendizaje colaborativo. Tesis doctoral. Universidad de Granada. (2011)
20. E. Montero y J. Carsí. MDA y Desarrollo de Videojuegos. Proc. JISBD'11 (2011).
21. Castillo Sánchez, F. A., Ordoñez Chacón, Y. L.: Entorno computacional basado en juegos colaborativos para apoyar procesos de enseñanza-aprendizaje. Proyecto para optar al título de Ingeniero de Sistemas, dirigido por César A. Collazos Ordóñez y codirigido por Natalia Padilla Zea. Universidad del Cauca, Colombia (2010).

Implementación de una Solución Reutilizable para una Funcionalidad de Usabilidad

Francy D. Rodríguez[1], Silvia T. Acuña[2]

[1] Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo s/n, 28660 Boadilla del Monte, Madrid, España
fd.rodriguez@alumnos.upm.es
[2]Departamento de Ingeniería Informática, Universidad Autónoma de Madrid, Calle Francisco Tomás y Valiente 11, 28049, Madrid, España
silvia.acunna@uam.es

Resumen. La usabilidad es un atributo de calidad y un aspecto crítico en los sistemas de software. Se ha establecido que algunas de las recomendaciones para mejorar la usabilidad dadas desde el campo de la Interacción Persona Ordenador tienen impacto en el diseño de software. En este artículo presentamos la implementación de una solución reutilizable para realizar una funcionalidad de usabilidad con alto impacto en el diseño: Abortar Operación. Desarrollamos tres aplicaciones web como casos de estudio, incluimos esta funcionalidad de usabilidad y buscamos elementos comunes en las implementaciones. Encontramos escenarios de aplicación, responsabilidades, clases, métodos, atributos y trozos de código comunes en los tres desarrollos. Con base en estos hallazgos, proponemos elementos reutilizables para incorporar la funcionalidad de usabilidad en el análisis, diseño y programación. Formalizamos la solución como un patrón de diseño y patrones de programación en tres lenguajes: PHP 5, Java y Visual Basic .NET.

Keywords: Usabilidad del software; Abortar operación; Patrón de diseño; Patrón de programación

1 Introducción

La usabilidad es un atributo de calidad que contempla que usuarios concretos puedan usar un producto satisfactoriamente de forma efectiva y eficiente logrando objetivos específicos en un determinado contexto [1]. La usabilidad es un aspecto crítico en sistemas software interactivos [2], y genera importantes ahorros e incremento de utilidades [3] [4] [5], razones por las cuales cada vez se tiene más en cuenta en el desarrollo de software [6].

La usabilidad de un sistema ha sido abordada ampliamente desde el campo de la Interacción Persona Ordenador (IPO). Las técnicas pertenecientes al campo de la IPO permiten alcanzar un adecuado nivel de usabilidad en un sistema, pero no son fáciles de asimilar desde la Ingeniería de Software (IS).

En IS, la usabilidad era vista como un requisito no funcional relacionado únicamente con la interfaz gráfica, lo que contribuía a que se abordara en etapas

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 159-172, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

tardías del ciclo de desarrollo. Posteriormente, algunos autores han argumentado que los aspectos de usabilidad generan restricciones estáticas y dinámicas sobre los componentes de software [7], otros resaltan que la separación entre la capa de presentación y la capa de negocio no es suficiente para garantizar la usabilidad, y que es necesario utilizar otras tácticas en el diseño de software si se quiere obtener un producto usable [8]. En estudios más recientes se ha evaluado la relación entre la usabilidad y los requisitos funcionales, llegando a concluir que, algunas características que mejoran la usabilidad, tienen impacto directo en la funcionalidad del software y en su diseño [6] [9], razón por la cuál se deben abordar desde el inicio del ciclo de desarrollo.

Las características de usabilidad con impacto sobre el diseño son las que implican el desarrollo o construcción de cierta funcionalidad dentro del sistema para mejorar la interacción usuario-sistema, tales como Retroalimentación de progreso, Alertas, Deshacer global o Preferencias. Las características de usabilidad con impacto sobre el diseño requieren componentes de software específicos para cumplir con sus responsabilidades asociadas, no solamente modificaciones a nivel de interfaz de usuario.

En [6] las autoras presentan evidencia empírica de la relación entre usabilidad y diseño de software, identifican características funcionales de usabilidad (CFU) con alto impacto sobre el diseño y miden su impacto en aplicaciones reales. Las funcionalidades identificadas son producto de recomendaciones IPO, cada autor IPO identifica diferentes subtipos para una CFU, cada subtipo se denomina mecanismo de usabilidad (MU). Los MUs tienen un nombre indicativo de su funcionalidad, por ejemplo la CFU Deshacer/Cancelar esta compuesta por cuatro MUs: Deshacer global, Deshacer sobre un objeto específico, Abortar operación y Regresar. En [9] se presentan guías para educir la funcionalidad de MUs con impacto en el diseño.

En este artículo presentamos una solución reutilizable para implementar el MU Abortar Operación (AO). La funcionalidad AO está enfocada en permitir al usuario cancelar una operación, un comando o salir de la aplicación de una forma segura y predecible. Con base en los estudios previos [6] [9] que identifican funcionalidades de usabilidad con alto impacto en el diseño, y proporcionan guías de educción, llevamos a cabo la implementación de la funcionalidad de usabilidad en tres casos de estudio distintos pensando en la reutilización.

La solución que proponemos busca proporcionar a un desarrollador herramientas para incluir de forma efectiva, eficiente, con menos errores y al menor costo posible, la funcionalidad de usabilidad AO en un sistema software. Esta solución consta de varios artefactos[2]: un conjunto de escenarios de aplicación, un patrón de diseño y código que implementa el diseño en tres lenguajes de programación (PHP 5, Java y VB .NET). Hemos llamado a la unión de diseño y código, patrón de programación. Los patrones de programación son patrones de bajo nivel de abstracción que implican el desarrollo de partes de componentes o de relaciones entre ellos [10]. Los patrones de programación son una solución por sí mismos y describen cómo implementar aspectos particulares de un patrón de diseño usando las características y potencialidades de un lenguaje de programación. A su vez, la solución reutilizable

[2] Un artefacto software es un objeto o producto de cualquier etapa del ciclo de desarrollo de software, algunos ejemplos son: casos de uso, diagramas de diseño, prototipos y librerías.

propuesta ha sido usada en el desarrollo de dos casos de estudio adicionales por dos desarrolladores diferentes, las implementaciones fueron hechas usando dos de los lenguajes para los cuales se presenta esta solución (Java y VB .NET).

Este artículo se ha estructurado de la siguiente manera: en la sección 2 se analizan trabajos relacionados que abordan la usabilidad mediante patrones. En la sección 3 se describe el método de investigación utilizado, detallando los casos de estudio desarrollados. En la sección 4 analizamos el proceso llevado a cabo para la identificación de escenarios de aplicación. En la sección 5 mostramos en detalle el proceso de formalización de la solución como un patrón de programación. En la sección 6 se hace referencia a la evaluación de la solución propuesta en dos casos de estudio adicionales. En la sección 7 se realiza una discusión acerca de las características de la solución propuesta. Finalmente en la sección 8 se presentan las conclusiones.

2 Trabajos relacionados

En los últimos años se han desarrollado estudios que buscan abordar los aspectos de usabilidad desde las primeras etapas del ciclo de desarrollo de software. En [11] [12] [8] los autores identificaron un conjunto de escenarios de usabilidad para los que la estrategia de separación de la interfaz de usuario no era suficiente para obtener un sistema usable, y definieron patrones arquitectónicos de soporte a la usabilidad. Posteriormente, se hizo un estudio a nivel empresarial usando estos patrones arquitectónicos, y con base en los resultados propusieron un Lenguaje de Patrones basado en Responsabilidades para patrones arquitectónicos de soporte a la usabilidad [13]. Este resultado constituye una descripción general de las responsabilidades que los distintos elementos funcionales deben cumplir, pero no proponen soluciones a un bajo nivel de abstracción para la implementación de los aspectos de usabilidad.

En línea con los trabajos anteriores se encuentra el proyecto STATUS [14], en el que se estudió la relación entre la arquitectura de software y la usabilidad, y se presentó una aproximación para mejorar la usabilidad aplicando un proceso específico de diseño. El proyecto STATUS plantea la incorporación de la usabilidad en momentos tempranos del desarrollo y adelanta el ciclo de evaluación/mejora al momento arquitectónico. Otro trabajo relacionado es en el que se proponen guías para la educción de funcionalidades de usabilidad [9] que apoyan la inclusión de la usabilidad desde la primera fase de desarrollo de software, en concreto desde la definición de requisitos.

Todos estos trabajos proporcionan herramientas para las primeras fases del proceso de desarrollo, pero no para las fases de diseño detallado e implementación. En [15] se presenta una propuesta que llega al nivel de implementación para la funcionalidad de usabilidad Alertas desde un enfoque diferente, la Programación Orientada a Aspectos. Esta propuesta busca solucionar un problema que se presenta con las aproximaciones expuestas anteriormente: el entrelazamiento de la funcionalidad propia de una aplicación con la funcionalidad de usabilidad. Sin embargo, todavía es necesario determinar cuales características de usabilidad pueden ser modeladas como aspectos [16] y evaluar las ventajas de usar este enfoque para la implementación de funcionalidades de usabilidad.

En este trabajo también llegamos a nivel de implementación, utilizando el enfoque de diseño y programación orientada a objetos. La solución reutilizable que se plantea, permite incluir la funcionalidad de usabilidad AO en un sistema software. Aunque se presenta entrelazamiento entre la funcionalidad de usabilidad y la del sistema, se proporcionan componentes reutilizables que encapsulan la funcionalidad del MU AO y se establecen los puntos de enlace con la aplicación. La solución ofrece las ventajas asociadas a la reutilización, tales como: menos errores, fiabilidad (código ya probado), adaptación a cambios en los requerimientos y menor coste de desarrollo.

3 Método de investigación

Para este estudio hemos utilizado un método de investigación inductivo en tres fases, implementando casos de estudio e induciendo a partir de ellos una solución general. Seleccionamos un MU con alto impacto en el diseño. La funcionalidad escogida fue AO que pertenece a la CFU Deshacer/Cancelar. Los criterios para la elección del MU fueron:

- Nivel de impacto sobre el diseño en cuanto a número de funcionalidades afectadas, determinado por las características de los casos de estudio a desarrollar.
- Facilidad de reconocimiento por parte de un usuario cuando use el sistema. Algunos MUs requieren la interacción del usuario para activar su funcionalidad por ejemplo retroalimentación de progreso. Otros MUs requieren que se produzca algún evento en el sistema como es el caso de Alertas, mientras que otros, como AO son opciones visibles para el usuario en todo momento.
- Facilidad de evaluación desde el punto de vista de las recomendaciones IPO. Todos los MUs tienen explicadas las recomendaciones IPO en las que se basan [17], por lo tanto, al final no fue un criterio decisivo para escoger el MU a desarrollar.

Los tres casos de estudio desarrollados son aplicaciones web interactivas basadas en requisitos para sistemas reales. El primero es un sistema de Administración de Indicadores que permite crear indicadores y datos simples, clasificar, consultar e importar datos. Es un sistema hecho en PHP 5[3] con una base de datos MySql. El segundo caso de estudio es un sistema web para generación de variables de pago, permite actualizar y gestionar información para pagos de nómina, calcula información de horas extras, nocturnas, festivas, y días trabajados. El sistema está hecho en VB .NET con una base de datos en Microsoft SQL. El tercer caso de estudio es un sistema de venta de comida saludable. El sistema permite suscripciones, crea y mantiene datos sobre del estado de salud del suscriptor, propone una dieta saludable, y da diferentes opciones para su compra y entrega. El sistema está hecho en Java con una base de datos PostgreSQL.

En la primera fase de la investigación se construyeron los sistemas web asegurando que además de su funcionalidad propia cumplieran con la funcionalidad asociada al

[3] Aunque PHP tradicionalmente no ha sido un lenguaje orientado a objetos, las últimas versiones del lenguaje ofrecen las funcionalidades necesarias para programar con este enfoque, usando clases y objetos.

MU AO. La educción de requisitos se realizó usando la guía de educción para este MU que se puede ver en el Apéndice Web[4]. En cada artefacto generado durante el proceso de desarrollo se marcaron los elementos relacionados con la funcionalidad de usabilidad. En la segunda fase identificamos los elementos comunes en las implementaciones del MU en los tres casos de estudio, y establecimos cuáles podían ser reutilizables. Al analizar la implementación de los tres casos de estudio surgió un elevado número de posibles escenarios de aplicación, cuya identificación no fue evidente al usar la guía de educción. La identificación y documentación de los escenarios solo fue posible a posteriori. Los escenarios son una parte de la solución que resulta ser útil para las fases de educción de requisitos, análisis y diseño.

En la tercera fase del estudio, formalizamos los resultados en forma de patrones. Se propone un diseño único y se adecuan las implementaciones en los tres lenguajes de programación utilizados: PHP 5, VB .NET y Java. La unión del diseño propuesto y los códigos implementados se formalizan como patrones de programación. Como parte de la tercera fase también se extraen "piezas de código" comunes, como un primer paso para la construcción de una biblioteca de componentes para la funcionalidad de usabilidad. Los resultados han sido utilizados por desarrolladores diferentes en casos de estudios independientes para evaluar su aplicabilidad.

4 Relación entre la Funcionalidad de Usabilidad y sus Múltiples Escenarios

La guía de educción del MU AO divide las preguntas en tres niveles: aplicación, operación y comando. A nivel de aplicación, la guía indica que se debe preguntar al usuario si es necesaria una opción para salir de la aplicación, y si es así, cómo se mostrará la opción al usuario. Según la recomendación IPO asociada, la opción de salir debe ser inmediata y obvia, incluso con uso de diálogos modales. Si la opción de salir se solicita cuando hay datos modificados, la opción "Salvar" debe estar presente.

El nivel de operación del MU AO se refiere a acciones que implican la ejecución de uno o varios pasos dentro de una aplicación, cada uno de los cuales requiere interacción con el usuario. Cada acción tiene como consecuencia la modificación del estado de la aplicación, ya sea por modificación de información de la base de datos, cambios en parámetros de configuración, o cambios en las variables de aplicación o de sesión en el caso de aplicaciones Web. Finalmente, el nivel de comando del MU AO se refiere a una instrucción que el usuario da a la aplicación a través de una única interacción, a través de botón, un enlace, eligiendo una opción de menú o cualquier otra opción suministrada por la aplicación.

Después de implementar los casos de estudio analizamos elementos comunes en cada fase del desarrollo para la funcionalidad de usabilidad. Comenzamos analizando los casos de uso de cada aplicación que incluían el MU AO. En todos los casos la funcionalidad del MU estaba definida como un camino alternativo a la funcionalidad principal. Describimos cada camino alternativo asociado al MU usando diagramas de secuencia. Encontramos una relación entre las preguntas de la guía de educción, las posibles situaciones que se presentan en la aplicación, las interpretaciones para un

[4] http://www.grise.upm.es/sites/extras/7/Usability_Elicitation_Pattern_AO.pdf

sistema web y el estado final del sistema. En el Apéndice Web[5] se encuentra una tabla resumen con las relaciones encontradas. En resumen, el comportamiento descrito en los diagramas de secuencia para la funcionalidad asociada al MU AO está definido por cuatro aspectos principales:

- Ubicación de la opción de cancelación (Botón cancelar en un cuadro de dialogo modal o no modal, botón cancelar en un formulario, selección de una opción diferente del menú mientras se realiza una operación, botón limpiar).
- Existencia o no de cambios pendientes por guardar en la aplicación.
- Intención de guardar o no los cambios existentes.
- Existencia o no de errores al momento de guardar.

Por ejemplo, a nivel de aplicación cuando se responde a las pregunta de la guía: ¿Es necesaria una opción para salir de la aplicación? Si la respuesta es positiva, y se atiende a la recomendación IPO de que debe existir la opción de guardar los cambios pendientes, se pueden presentar dos situaciones, que al solicitar salir existan o no cambios por guardar. Si no existen cambios, el sistema pasará al estado siguiente. Si por el contrario, existen cambios por guardar, se debe preguntar al usuario si desea o no guardarlos, si el usuario no desea guardar los cambios se pasa al estado siguiente, pero si el usuario desea guardar pueden darse varias situaciones: que se guarde con éxito, que existan errores de validación o que existan errores de base de datos, en cada caso la aplicación debe quedar en un estado predecible.

Para tener una lectura rápida de las posibles combinaciones para el MU AO construimos árboles de escenarios para cada nivel: aplicación, operación y comando. En la **Fig. 1** se presenta el árbol de escenarios a nivel de aplicación, a este nivel existe una única opción para salir de la aplicación, el árbol muestra la posible casuística que puede generarse dependiendo de si existen o no cambios por guardar, de la decisión del usuario de guardar o no los cambios, y del resultado del proceso de guardado. La parte derecha del árbol muestra las posibles respuestas del sistema y el estado final de la aplicación.

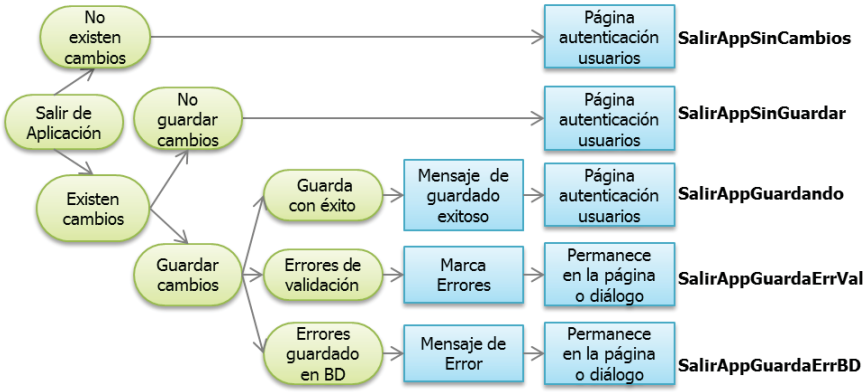


Fig. 1. Descomposición en escenarios del MU Abortar Operación a nivel de Aplicación

[5] http://www.grise.upm.es/sites/extras/7/Relacion_Guia_Arbol_MU_AO.pdf

A cada rama del árbol le dimos un nombre y la describimos usando diagramas de secuencia, por ejemplo, al escenario en que se solicita salir de la aplicación, existen cambios pendientes, se desea guardar los cambios pero se presenta un error de validación, lo llamamos *SalirAppGuardarErrVal* (**Fig. 1)**. El diagrama de secuencia asociado se puede ver en la **Fig. 2**.

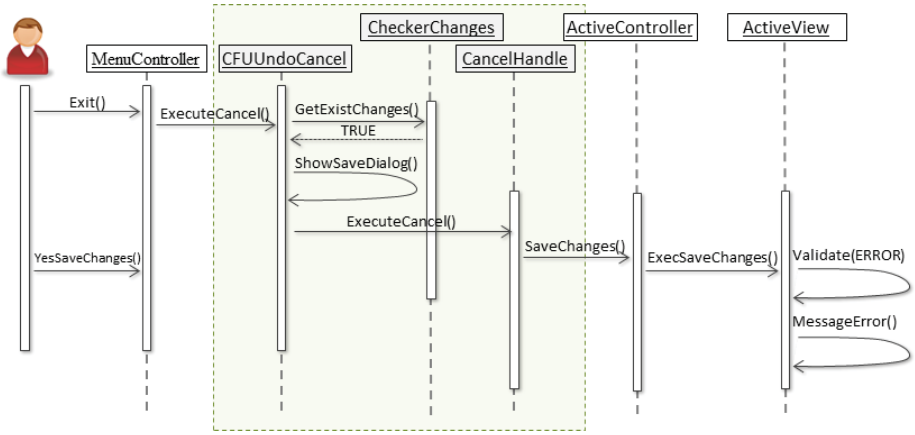


Fig. 2. Diagrama de secuencia para el escenario SalirAppGuardarErrVal

En el apéndice Web[7] se encuentran los tres árboles de escenarios y los diagramas de secuencia para cada rama. Obtuvimos en total 22 escenarios distintos. Nótese que a nivel de aplicación obtuvimos 5 escenarios, pero a nivel de operación el número de escenarios ascendió a 16, principalmente porque a diferencia del nivel de aplicación que solo tiene un posible origen (opción salir), a nivel de operación encontramos cuatro posibles orígenes: botón cancelar en una caja de dialogo, botón cancelar en un formulario, selección de otra opción de la aplicación y botón limpiar.

Durante la elaboración de los diagramas de secuencia identificamos un conjunto de responsabilidades asociadas a la funcionalidad del MU. Se requiere:

- Escuchar las acciones del usuario para determinar cuándo se solicita salir de la aplicación, cancelar una operación o cancelar un comando.
- Conocer en todo momento si existen o no cambios por guardar.
- En caso de que existan cambios en la aplicación, preguntar al usuario si desea o no guardar estos cambios y conocer la acción a seguir dependiendo de la respuesta recibida.
- Conocer cuál es el estado anterior y actual de la aplicación.
- Conocer como guardar los cambios pendientes de la aplicación independientemente de la operación o comando que se esté ejecutando.

Para cubrir las responsabilidades identificadas se definió un conjunto de componentes, que se pueden ver en la **Tabla 1** y que son usados en los diagramas de secuencia. En la **Fig. 2**. se ven sombreados los tres componentes relacionados con la funcionalidad de usabilidad.

[7] http://www.grise.upm.es/sites/extras/7/Escenarios_MU_AO.pdf

Tabla 1. Responsabilidades de los componentes del MU Abortar Operación

Componente	*Responsabilidad*
CheckerChanges	Mantiene e informa si existen cambios por guardar en la aplicación
CancelHandle	Guarda cambios en caso de que se aborten operaciones y deja al sistema en el estado adecuado (predecible y seguro para el usuario).
CFUUndoCancel	Recibe la solicitud de abortar operación (salir o cancelar), consulta al componente CheckerChanges si existen cambios, pregunta al usuario si desea guardar cambios y llama al método adecuado en cada caso.
HistorySteps	Mantiene y proporciona información del estado anterior y el estado actual del sistema.

Para generalizar los diagramas de secuencia usamos dos patrones reconocidos: el Patrón Modelo, Vista, Controlador y el patrón Fachada. La fachada se usa como punto de entrada a la funcionalidad de usabilidad, la vista se refiere a la interfaz con el usuario, el controlador recibe los eventos de usuario y hace las solicitudes a los componentes correspondientes, y el modelo maneja las reglas de negocio.

El conjunto de escenarios identificados describe toda la funcionalidad descubierta para el MU AO en los tres casos de estudio. Concluimos que el uso de la guía de educción resulta aún demasiado general para acometer la implementación de la funcionalidad de usabilidad. Al realizar las preguntas de la guía quedan demasiadas opciones abiertas sobre las que el desarrollador debe decidir, lo que significa posibles omisiones de escenarios de aplicación y demoras y/o errores en el diseño o implementación. El uso de los árboles de escenarios y de la descripción de cada escenario a través de diagramas de secuencia es una herramienta útil desde el momento de la educción de requisitos.

5 Solución Reutilizable Propuesta para la Funcionalidad de Usabilidad Abortar Operación

En esta sección describimos el proceso usado para obtener elementos reutilizables a nivel de diseño e implementación. Además de los escenarios de aplicación descritos en la sección 4, en los desarrollos finales de los tres casos de estudio encontramos coincidencias entre atributos, métodos y clases. A nivel de clases, encontramos que:

- Los tres diseños tienen una clase que funciona como fachada. Consecuencia de que los tres casos se construyeron pensando en ser soluciones reutilizables.
- En los tres casos existe una clase que tiene la responsabilidad de conocer si existen cambios pendientes. La clase tiene un solo atributo y tres métodos con funcionalidad similar. El atributo permite saber si hay cambios pendientes, y los tres métodos permiten modificar y consultar el valor del atributo.
- En los tres casos existe una clase que encapsula los métodos principales para responder a una solicitud, ya sea salir de la aplicación, o cancelar una operación o comando. Conoce cómo guardar cambios pendientes y cuál es el estado en el que debe quedar el sistema después de la solicitud.
- Hay otra clase que aparece en dos de los tres casos de estudio, encargada de mantener la información del estado anterior y actual del sistema. En el caso de VB

.NET no existe una clase con éste propósito porque utiliza características propias de la tecnología.

A nivel de atributos, encontramos que podían variar según la tecnología usada, pero también coincidían en varios puntos. Por ejemplo, en la clase que encapsula los métodos principales para atender una petición de AO, en los tres casos de estudio se usa un atributo para mantener la información necesaria para salvar los cambios, aunque se implementa de forma distinta en cada uno. Lo mismo sucede con la instrucción para cerrar un cuadro de dialogo o para salir de la aplicación.

También se encontraron coincidencias en otro tipo de decisiones de diseño, es el caso del uso de una fachada (CFUUndoCancel) definida como un Singleton, debido a que varios atributos en la solución debían ser únicos por sesión. El patrón Singleton asegura que solo existe una instancia de la clase y por lo tanto una única vía para actualizar sus datos. En el caso del MU AO hay datos que deben ser únicos por sesión: la existencia o no de cambios pendientes por guardar, el último estado de la aplicación, la forma de salvar los cambios y cuál es el cuadro de dialogo activo.

La principal diferencia que encontramos fue el manejo de los estados del sistema. En el caso de Java, por ejemplo, se utiliza una clase a nivel de servidor para mantener la información del estado anterior y actual del sistema, debido principalmente a la tecnología usada (JavaServerFaces). Mientras que en con VB .NET y PHP 5 no se usaron clases, sino variables de sesión.

Al finalizar el análisis de los tres diseños concluimos que muchos atributos, métodos y clases cumplían con las mismas responsabilidades y por lo tanto se podían unificar en un solo diseño que se puede ver en la **Fig. 3**. En cuanto a los elementos diferentes no son excluyentes, por el contrario se complementan para plantear un diseño que cubra todos los escenarios de aplicación descubiertos.

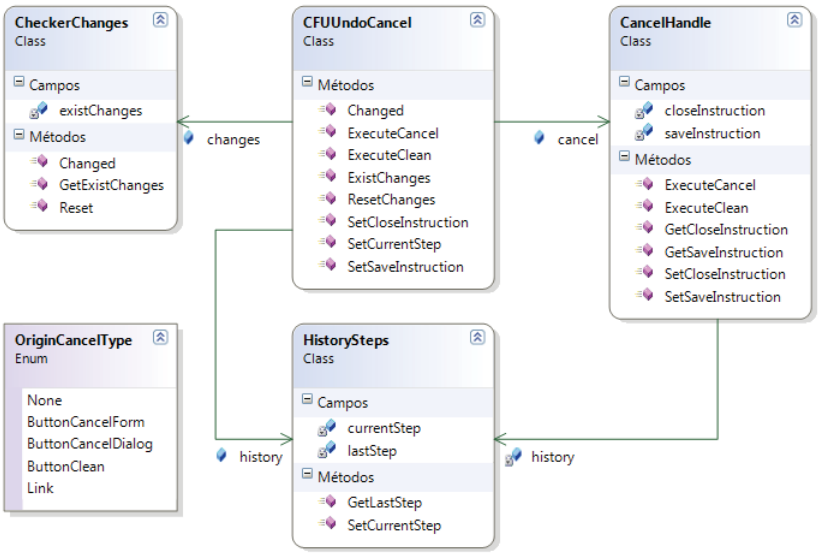


Fig. 3. Diagrama de clases unificado para el MU Abortar Operación

A nivel de programación, en los tres casos de estudio se encontró que una proporción significativa de la lógica está en el cliente. Al ser sistemas web, la implementación a nivel de cliente esta en el mismo lenguaje script para los tres casos: Javascript. Se hicieron algunas modificaciones para unificar los tres códigos script y se obtuvo un código único para los tres sistemas, que cubre todos los escenarios encontrados. En la **Tabla 2** se puede observar el código de uno de los componentes comunes: Checker Changes. En cuanto al código de la parte servidor, las diferencias están dadas por la tecnología utilizada, el diseño se modifica para adecuarse a la tecnología, aunque los componentes tengan las mismas responsabilidades.

Tenemos entonces una propuesta de diseño único y las implementaciones realizadas en tres lenguajes de programación distintos. La unión entre el diseño y la implementación la hemos formalizado como un patrón de programación. En el apéndice Web mostramos los patrones de programación para el MU Abort Operation en VB .NET[11], Java[12] y PHP 5[13].

Tabla 2. Código de la clase que mantiene información de si existen o no cambios pendientes

```
//Class: CheckerChanges
//Description: Mantiene información de cambios
function CheckerChanges(){
    this.ExistChanges = false; }
CheckerChanges.prototype.Changed = function() {
    this.ExistChanges = true; }
CheckerChanges.prototype.Reset = function() {
    this.ExistChanges = false;}
CheckerChanges.prototype.GetExistChanges = function()
{ return this.ExistChanges;}
```

Los patrones de programación que proponemos cubren todos los escenarios descubiertos para el MU AO, sabemos que en nuevas implementaciones aplicarán subconjuntos de ellos, razón por la cual no siempre se usará todo el código implementado. El código ejemplo asociado a los patrones se ha documentado para que sea fácil escoger las partes que sean útiles a un desarrollador dependiendo del alcance de la funcionalidad del MU que necesite implementar.

6 Evaluación

La solución propuesta ha sido aplicada en dos casos de estudio adicionales elaborados por diferentes desarrolladores con experiencia en programación. Los desarrolladores elaboraron los casos de estudio como parte de su Trabajo de Fin de Máster en la Universidad Politécnica de Madrid. Los casos de estudio fueron diferentes entre si, basados en requisitos reales para sistemas web y usando lenguajes de programación distintos: Java y VB .NET. Para la educción de requisitos los desarrolladores usaron

[11] http://www.grise.upm.es/sites/extras/7/PP_AO_VB_NET.pdf
[12] http://www.grise.upm.es/sites/extras/7/PP_AO_Java.pdf
[13] http://www.grise.upm.es/sites/extras/7/PP_AO_PHP.pdf

las mismas guías de educción que se usaron en este estudio [9] [17], pero adicionalmente utilizaron los artefactos reutilizables resultado de este trabajo:

- Escenarios de aplicación. En la educción de requisitos constituyen una guía para incluir escenarios que no son evidentes al usar la guía de educción, y evitar que se quede por fuera alguno de ellos a la hora de hablar con los usuarios. En el análisis sirven para entender la funcionalidad de usabilidad y como interactúa con el resto del sistema. Permite estimar la complejidad que se agregará al sistema.
- Patrón de diseño. Proporciona una descripción de los componentes requeridos para cumplir con las responsabilidades asociadas al Mecanismo de Usabilidad.
- Patrones de programación. Muestran una solución real utilizando el patrón de diseño con una tecnología específica. Proporcionan trozos de código reutilizables.

Según los desarrolladores, mediante un cuestionario de validación de la propuesta, concluyen que a pesar del tiempo que invirtieron inicialmente para entender la solución, el resultado final fue positivo, ya que los sistemas implementados cumplen con la funcionalidad de usabilidad, y que usarán la solución en futuras implementaciones.

Algunas de las cuestiones planteadas tenían que ver con la cantidad de esfuerzo adicional requerido en cada fase del desarrollo para incorporar la funcionalidad de usabilidad usando los patrones propuestos. En este sentido los desarrolladores encontraron que el uso de los patrones requiere en primera instancia un tiempo adicional considerable tanto para entender su funcionamiento como para incorporar los elementos a la solución. Los datos de esfuerzo variaron entre los desarrolladores debido entre otras cosas a que usaron diferentes modelos de desarrollo, uno usó el modelo tradicional en cascada y otro el modelo de desarrollo incremental.

También se pidió a los desarrolladores que propusiesen posibles mejoras a la solución y detallaran sus impresiones en cada momento del proceso de desarrollo. Se plantearon varias sugerencias entre las que se encuentran: mejor documentación del código y creación de aplicaciones demo, también propusieron e implementaron soluciones alternativas para el mismo diseño. Otras de sus conclusiones fueron: los escenarios son útiles y fáciles de usar una vez se entiende su propósito, tanto como complemento a las guías de educción, como en la etapa de análisis. El patrón de diseño fue el más sencillo de usar y fue aplicado al 100%.

Los desarrolladores también detallaron la forma en que usaron el código suministrado. Para la parte cliente se presentaron dos situaciones, una en la que el código fue usado como caja negra, es decir, pudieron usar el código sin modificarlo, y otra, en la que fue necesario hacer pequeñas adaptaciones. Las modificaciones fueron necesarias por incompatibilidades entre las tecnologías o versiones del lenguaje. En cuanto a la parte del servidor, no fue posible usar el código como caja negra, sin embargo, sí se pudo usar un esquema de copiar/pegar.

7 Discusión

Después de implementar el MU AO en los tres casos de estudio, encontramos que existían múltiples escenarios de aplicación dependiendo de las respuestas de los usuarios a las guías de educción. Debido a que las tres tecnologías utilizadas para

implementar los casos de estudio son orientadas a objetos, se usan clases, métodos y atributos, lo que permitió tener elementos similares de comparación. Durante el desarrollo de los casos de estudio se marcaron los elementos que tenían que ver con la funcionalidad de usabilidad lo que también facilitó la búsqueda de similitudes.

Como se ha expuesto en las secciones 4 y 5, el análisis de la funcionalidad del MU comienza identificando los escenarios de aplicación, al unir todos los escenarios de aplicación obtuvimos una descripción detallada de la funcionalidad que debía cubrir la solución reutilizable que fuésemos a proponer.

Del análisis de las responsabilidades necesarias para cumplir con la funcionalidad de usabilidad en cada caso estudio, resultaron un conjunto de responsabilidades comunes para cubrir todos los escenarios de aplicación del MU. Se definieron componentes para cumplir con estas responsabilidades. En algunos casos, el resultado fue un componente para cubrir una responsabilidad y en otros casos, un componente para dos o tres responsabilidades similares. Con base en los componentes identificados se propone un diseño que busca ser reutilizable. El diseño encapsula el conocimiento necesario para cubrir cualquiera de los escenarios del MU.

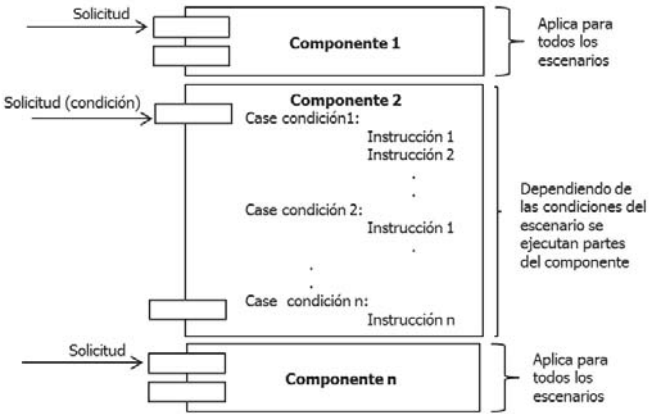


Fig. 4. Estructura de los componentes propuestos para el MU Abortar Operación

Cada componente se diseñó como una clase. La funcionalidad de un escenario no es cubierta por un componente específico, si no por la interacción de varios componentes y la interacción esta descrita por el diseño de clases. Como se muestra en la **Fig. 4** en algunos de los componentes no fue necesario hacer ninguna diferenciación relacionada con condiciones dadas por los escenarios, el código cumple con su responsabilidad asociada sin hacer referencia a los escenarios. En otros casos fue necesario hacer diferenciaciones según algunas condiciones dadas por los escenarios por lo tanto existirán partes del código que no se ejecuten.

Debido a que los sistemas software desarrollados son aplicaciones web, algunos de los componentes identificados son para la parte cliente y otros para la parte servidor. La parte del cliente se implementó usando JavaScript en los tres casos de estudio dando como resultado código similar y comparable, pero la implementación de los componentes de la parte servidor dependen del lenguaje y su comparación llego solo a nivel de diseño. Las “piezas de código”, que son iguales en los tres casos,

corresponden a la capa cliente, están implementadas en lenguaje JavaScript y cubren todos los escenarios documentados. La encapsulamos en un solo archivo y consideramos que es un primer paso para la construcción de una biblioteca de componentes para la usabilidad.

8 Conclusiones

En este artículo, detallamos la obtención y validación de una solución reutilizable para la implementación de la funcionalidad de usabilidad Abortar Operación. A través de implementaciones reales encontramos que existen elementos comunes que pueden generalizarse como artefactos reutilizables para diferentes fases del proceso de desarrollo. Los resultados de este estudio están limitados a aplicaciones web altamente interactivas y desarrolladas bajo el paradigma orientado a objetos.

La funcionalidad que abarca la solución reutilizable está limitada a los escenarios de aplicación encontrados en los casos de estudio, es posible que al desarrollar nuevos casos de estudio surjan nuevos escenarios de aplicación. La documentación de los escenarios a través de diagramas de secuencia es una herramienta útil desde el inicio del ciclo de desarrollo. En la fase de educción y especificación de requisitos permite verificar que se tengan en cuenta todos los casos posibles en los que puede aplicar la funcionalidad de usabilidad. En la fase de análisis permite evaluar la forma en que se verán afectadas las funcionalidades del sistema software y proporciona una idea clara de cómo realizar su implementación.

El patrón de diseño propuesto encapsula la funcionalidad necesaria para cubrir con las responsabilidades descritas en los escenarios de aplicación. Será necesario modificar el diseño según la tecnología en que se implemente, aunque encontramos que el código para la parte cliente puede ser común a cualquier aplicación web. Los patrones de programación serán más útiles cuando se usen las mismas versiones del lenguaje de programación. Formalizar los resultados como un patrón de diseño permite describir la solución a nivel más general y los patrones de programación proporcionan código útil o en su defecto una guía para su implementación en otros lenguajes de programación.

La aplicación en otros casos de estudio llevada a cabo por desarrolladores distintos, permitió encontrar fallos en la documentación y la necesidad de proporcionar aplicaciones demo adicionales al código. También permitió observar que muchos de los artefactos reutilizables proporcionados fueron útiles y que aunque en una primera implementación se requiera un tiempo adicional para el entendimiento y aprendizaje del patrón, son potencialmente reutilizables en nuevas implementaciones. Como es inherente a la definición de patrón, este estará en mejoramiento continuo. En cada nueva implementación se podrán incluir nuevas funcionalidades, mejoras al diseño y código en otros lenguajes o versiones.

Como continuación de este trabajo, aplicaremos el mismo método expuesto para obtener soluciones reutilizables para otras funcionalidades de usabilidad con alto impacto en el diseño. Buscaremos obtener un repositorio de patrones y componentes para facilitar la implementación de funcionalidades de usabilidad y/o establecer un procedimiento para facilitar su obtención.

Agradecimientos. Este trabajo ha sido financiado por el Ministerio de Ciencia e Innovación de España a través de los proyectos Tecnologías para la Replicación y Síntesis de Experimentos en IS (TIN2011-23216) y Go Lite (TIN2011-24139).

Referencias

1. ISO Std. 9241-11: Ergonomic Requirements for Office Work with Visual Display Terminals. Part 11: Guidance on Usability (1998)
2. Constantine, L., Lockwood, L.: Software for use: A practical Guide to the Models and Methods of Usage-centered Design. Addison Wesley (1999)
3. Donahue, G. M.: Usability and the Bottom Line. IEEE Software, 18, 22-30 (2001)
4. Nielsen, J.: Return on Investment for Usability. Alertbox. http://www.useit.com (2003)
5. Chrush, M.: The Whiteboard: Seven Great Myths of Usability. Interactions 7, 13-16 (2000)
6. Juristo, N., Moreno, A. M., Sanchez-Segura, M.: Analysing the impact of usability on software design. J. System and Software 80:9, 1506-1516 (2007)
7. Perry, D., Wolf, A.: Foundations for the study of software architecture. ACM Software Engineering Notes 17:4, 40-52 (1992)
8. Bass, L., John, B.E.: Linking usability to software architecture patterns through general scenarios. Journal of Systems and Software 66:3, 187-197 (2003)
9. Juristo, N., Moreno, A. M., Sanchez-Segura, M.: Guidelines for Eliciting Usability Functionalities Software Engineering. IEEE Transactions on Software Engineering 33, 744-758 (2007)
10. Bushmann, F., Meunier, R., Rohnert, H.,Sommerlad, P.: Pattern - Oriented Software Arquitecture. A system of patterns. John Wiley & sons Ltd., England (1996)
11. Bass, L., Jhon, B.E.: Supporting usability through software arquitecture. IEEE Computer 34(10), 113-115 (2001)
12. Bass, L., Bonnie, E., Kates, J.: Achieving Usability through Software Architecture. Technical Report CMU/SEI-2001-TR-005. Software Eng. Inst., Carnegie Mellon Univ. (2001)
13. Bonnie E., Bass, L., Golden, E., Stoll, P.: A Responsibility-Based Pattern Language for Usability-Supporting Architectural Patterns. Proceedings of EICS. Pittsburgh, PA, USA (2009)
14. STATUS Project. Software Architecture that supports Usability, http://www.grise.upm.es/rearviewmirror/projects/status/index.html (2004)
15. Campos G., Acuña, S. T., Macías, J. A.: Implementación de Propiedades de Usabilidad con Impacto en el Diseño Mediante la Programación Orientada a Aspectos. Actas del XI Congreso Internacional de Interacción Persona-Ordenador, Valencia, España (2010)
16. Pinto, M., Fuentes, L.: Aspect-Oriented Modeling of Quality Attributes. ECSA, 2008. LNCS Vol. 5292, pp. 334-337 (2008)
17. Juristo, N., Moreno, A.,Sanchez-Segura, M.: Usability Elicitation Patterns (USEPs), http://www.grise.upm.es/sites/extras/2/ (2006)

An experience migrating a Cairngorm based Rich Internet Application from Flex to HTML5

Juan A. Pereira, Silvia Sanz, Inko Perurena, Julián Gutiérrez, and Imanol Luengo

UPV/EHU, Donostia, SPAIN,
{juanan.pereira,silvia.sanz,inko.perurena,julian.gutierrez}@ehu.es
iluengo004@ikasle.ehu.es

Abstract. This paper shows a new approach to migrate a Flex based Rich Internet Application (RIA) that is using the Cairngorm architectural framework to HTML5. The migration has been done rewriting the Cairngorm code, from ActionScript to Javascript and using the result as super classes that have to be implemented by concrete Javascript classes. The similarities between the original ActionScript and Flex code and the resulting Javascript code help the developers in the migration process. To overcome the problems that arise due to the fact that some multimedia features are not yet implemented in any browser -despite the HTML5 specification states that they will be in the future -, we have suggested the use of minimal Flash widgets that communicate with their HTML Wrapper page by means of the ExternalInterface API. Doing so, it will be easy to replace these widgets with HTML5 objects whenever they are implemented by major browsers.

Keywords: Cairngorm framework, Flash, Flex, HTML5, migration, RIA

1 Introduction

HTML5 has evolved a lot since 2010, when Fraternali [8] stated that "for HTML5 to assume a central role in RIA development", it must ensure that "the complex set of features embodying the new language have internal coherence and meet RIA developers' requirements". The waiting for that central role is almost over, and that has led to strong movements in the RIA development market. In its official blog, Adobe recently announced that they were to axe the development of Flash Player for mobiles. Adobe will commit their efforts to improve development tools for HTML5 application building because in the long-term, they "believe HTML5 will be the best technology for enterprise application development" (see [1])

HTML5 related technologies (comprising HTML, JavaScript and CSS) are becoming increasingly capable of hosting RIA applications. Advances in expressiveness (e.g. Canvas), performance (e.g. VM and GPU acceleration in many browsers) and application-related capabilities (e.g. offline storage, web workers) will continue to evolve at a rapid pace. According to Adobe, in 3-5 years from

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 173-184, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

now HTML5 might support the majority of use cases where Flash/Flex is used today. A strong and educated opinion being Flash a major player in the RIA development arena (with a huge market penetration when comparing with the alternatives: Adobe Flash 96%, JavaFX 76%, Microsoft Silverlight 68%, as the StatsOwl website shows [15])

2 From Flex to HTML5

In this paper, we promote the idea that it is convenient to migrate as soon as possible to an HTML5 environment to take advantage of the broad adoption of this technology and be prepared for a complete migration when all browsers implement the specifications fully. It is not merely a refactoring, because we do not want to "factor out" new abstractions by applying small changes to the source code of an application that preserves its behaviour ([12]). We are proposing a whole re-engineering of the code - but maintaining the architecture and internal classes' responsibilities - , that allows us to migrate a Flex-Cairngorm based RIA called Babelium to a combination of HTML5, Javascript and CSS. Flex offers an application framework built upon ActionScript code to develop Rich Internet Applications. When compiled, Flex code generates a Flash binary object. Cairngorm is an implementation of several design patterns that form a lightweight architectural framework for Flex applications.

Of course, there are some model-driven development solutions that allow developers to build, refactor and reengineer RIA applications, (see [13] and [11]). Once applied, we achieve a good model for operating and reasoning upon. But yet, there is a final problem to solve: how to generate the original Flex's equivalent in HTML5 code? And what if the Flex application already follows a good architectural framework (as Cairngorm) and uses design patterns? Taking advantage of the fact that many Flex RIAs have been built using the Cairngorm framework, in this paper we propose a way to migrate such applications to an HTML5 environment.

To show the feasibility of this approach, this paper will describe how it has been applied to a real scenario, BabeliumProject.com, where a migration from Flex to HTML5 was needed.

There is no silver bullet. While the HTML5 standard specifies many multimedia related features, there is no browser that has fully implemented all of them. For example, as of this writing, it is impossible to stream or record webcam generated content using only HTML5, as the standard specifies ([17]), even though there are engineers working hard in the WebRTC implementation ([19]) to achieve this milestone. So it is compulsory to offer a temporary hybrid solution that mixes a pure HTML5 solution with Flash based widgets where there is no alternative.

3 Flash to HTML5 migraton strategies and applications

Currently, several applications exist offering various Flash to HTML5 migration capabilities. Their main objective is the automation of the process: having the source code of a Flash program or directly using the binary file (SWF format) they try to generate an equivalent one using only HTML5. The following overview briefly discusses these applications and their approaches.

3.1 Translating ActionScript to Javascript

Adobe Wallaby. "Wallaby" is the codename for an experimental technology that converts the artwork and animation contained in Adobe® Flash® Professional (FLA) files into HTML5. Although there are a lot of features missing ([18]), it is a good starting point for Flash designers that want to have a first scaffolding version of their work in HTML5.

Jangaroo. Jangaroo's compiler ([10]) translates a subset of ActionScript 3 into JavaScript 1.5, which is understood by all major browsers. It is an open source application that adds Javascript constructions to simulate a subset of ActionScript 3 features that are not natively available, like classes, packages, private members, and static typing.

FalconJS. FalconJS is an experimental MXML and ActionScript to HTML and JavaScript cross-compiler, built upon Falcon [7], the future compiler of Flex applications. Falcon parses ActionScript source code and creates an abstract syntax tree (AST), which gets reduced to byte code that can be interpreted by an ActionScript Virtual Machine (AVM2) like the Flash Player. FalconJS replaces the byte code generator with a JavaScript generating backend. As of today it is only an internal prototype but it seems to be the way to go for Adobe.

3.2 Interpreting Flash binary files (SWF) in Javascript

This approach consists of virtual machines implemented in JavaScript that interpret the same byte code as the Flash plug-in does. As of today, Google's Swiffy [16] and Gordon [9] simulate only a very old and limited version of the Flash player virtual machine (version 8).

3.3 Reverse engineering

Using Sothink's SWF to HTML5 reverse engineering tool, it is possible to generate the HTML5 equivalent of a SWF binary file. But, as of today it is only suitable for converting very simple SWF animations, without any interaction that requires Javascript.

None of the solutions take advantage of the fact that Flex applications might be written using an architectural framework. In those cases, we advocate for the idea that it will be very interesting and useful to have a similar framework already translated to HTML5, thus easing the process of the migration for the developers, that will feel the new environment very similar to the original Flex application.

4 A real scenario to migrate: Babelium Project

BabeliumProject.com is a rich internet application to practice oral language in a collaborative way, developed by the research group GHyM of the University of the Basque Country (UPV/EHU). Babelium allows users to display video-recorded conversations, usually involving 2 or 3 people, in Spanish, English and Basque in this beta stage.

The usual workflow is as follows: once viewed one of the videos, the user can choose to take the role of one of the characters and perform the associated dubbing (Figure 1). That is, after selecting the role you want to dub, the video will start playing until the time for the selected role comes, at which point Babelium will record the user's voice (via the microphone and, if the user wishes, also the image of her webcam). Just as in real life, you have a limited time-frame to respond, exactly the same amount of time taken by the role of character in the original video. When the user completes the dubbing of the conversation,

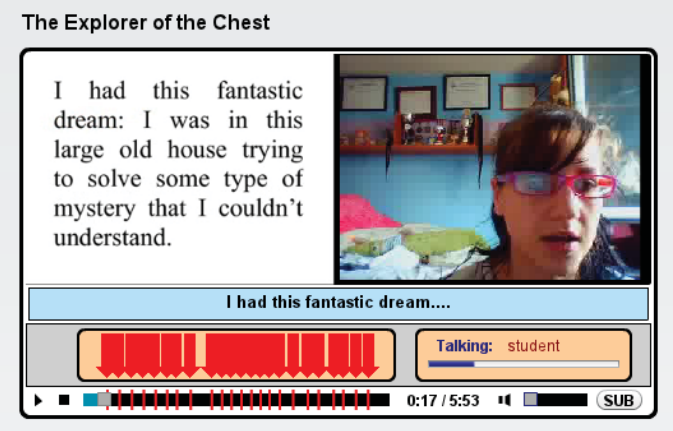


Fig. 1: A user practicing her oral skills with Babelium

she will have the chance of reviewing her work. When the user determines that the recording is ok, she will upload it to the server so that it will be available to be evaluated by other users. The evaluation is done collaboratively, e.g. , if the recording was done in Basque, Basque knowledgeable users will have the

chance to judge the user's performance. This assessment may also include text comments and/or video-comments.

The collection of videos available on Babelium covers various levels of difficulty, following the nomenclature of the European Framework of Reference for Languages (A1, A2, B1, B2, C1), used in several language schools.

Moreover, although Babelium has videos already available in Basque and English, anyone can upload their own creations, in any language.

On the other hand, besides serving as a tool for practicing oral language skills, the users can also use it to improve their listening comprehension of a language, using the "Subtitle" module. When a new video is uploaded to Babelium, users are able to subtitle it in a collaborative way, through an editor created for this job that is fully integrated into the application.

5 The Cairngorm Framework

In this paper, we mention two types of frameworks, application and architectural. Application frameworks (like Flex) provide a rich set of classes that allow developers to assemble interactive, easy-to-use applications, while architectural frameworks (like Cairngorm) "provide a defined structure that an application is built around, but generally do not add functionality to it" ([20])

Although there are several frameworks to help developers build complex, enterprise-oriented, Rich Internet Applications with Adobe Flex, including PureMVC, Mate and Clear Toolkit, Cairngorm is the only one officially endorsed by Adobe itself (see [3])

Cairngorm is an open source architectural framework for building applications using several design patterns such as Model View Controller (MVC), Command, and Delegate.

The data flow diagram between the architectural components of a Cairngorm based application is shown in Figure 2. Whenever the user interacts with the graphical interface of the application, Cairngorm layers dispatch a chain of events and commands to manage that interaction, starting from the View layer. This layer's role is to fire events and bind its components to the data stored in the Model. The values and states of these components are usually modeled with Value Objects and other attributes of the Model layer. For example, when the user clicks a button in the View layer, an event is dispatched, and that event is caught by the FrontController. This layer knows how to map all application events to their appropriate actions (Commands) and how to execute each one. This command execution will instantiate a Delegate object that accesses a certain service. These delegates become proxy objects that represent and control the access to remote services (using HTTPService, RemoteObject or WebService classes), and when called, will return the result or the fault to the invoker command. The last step is performed by the Command class, that will update the data in the model (typically, setting values in a collection of value objects). This, in turn, will update the widgets of the View layer binded to the variables that have changed in the model.

6 Migrating from Flex to HTML5

Our initial Flex application followed the architecture scheme of Figure 3. Two main parts are depicted: the Cairngorm based microarchitecture and the presentation layer which has two inner submodules, namely the View Component formed by MXML files and the Presentation Model, advocated for removing state and logic (both programmed in ActionScript) from MXML view components.

6.1 Migrating from Flex to HTML5: the client side

Our first migration proposal is focused in the Presentation Layer. In the following sections we will explain that the View Component can be easily translated to an HTML5+CSS3 design although the Presentation Model can be more problematic. Indeed, a combination of HTML5 + CSS3 + JavaScript events is needed to solve the translation. This implementation will be tightly coupled to a tailor made JavaScript based Cairngorm implementation. This has been done mimicking all the generic classes and interfaces that form the microarchitecture of Cairngorm to build a Cairngorm.js file. Once the JS microarchitecture is ready, all the AS and Flex code that uses it also ought to be translated to Javascript, in this case, forming a babelium.js file. There will sit all the specific Events, Commands and Delegates that implement the interfaces provided by Cairngorm.js. The similarities between the original ActionScript (AS) code of Babelium and the resulting JavaScript code, as seen in Figure 4 and Figure 5, help the developers to smooth the transition to HTML5.

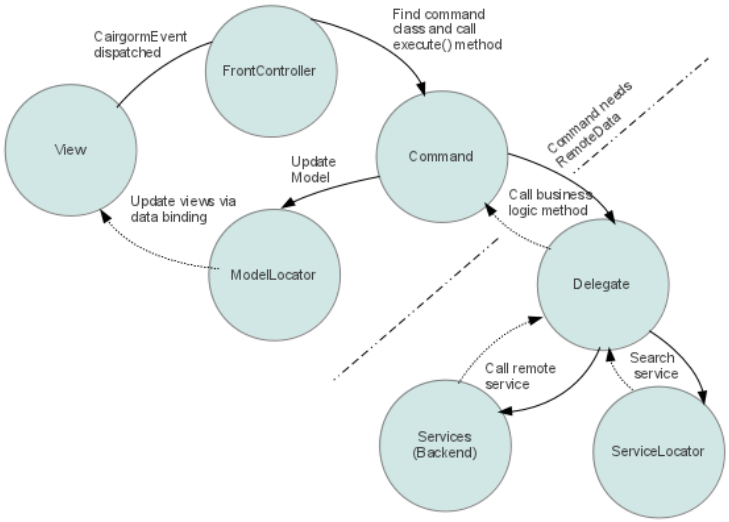


Fig. 2: Data flow between architectural components of Cairngorm framework

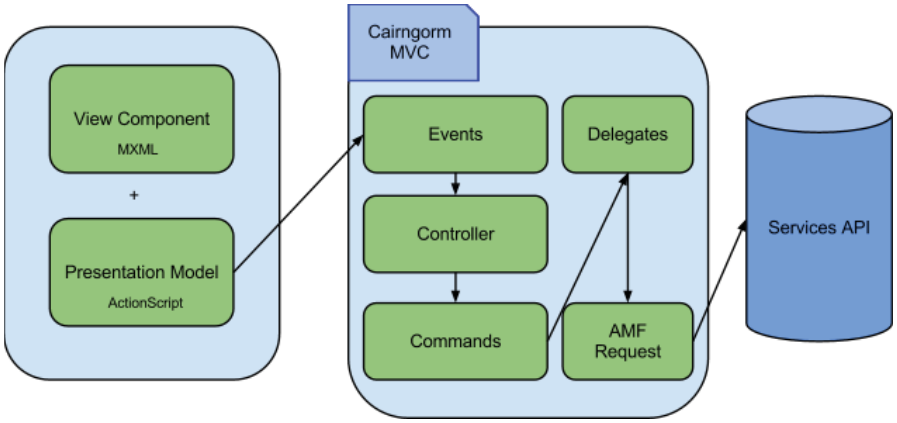


Fig. 3: Main blocks of a Cairgorm based application architecture

Between Cairngorm.js and babelium.js, we ended up having a great amount of JS classes. That hinted us to use a JavaScript optimizer to help us reducing the size of the application, hence, optimizing the user experience. Google's Closure ([4]) Compiler has proved to be a great solution for this task and compared to the other most used JavaScript optimizer tool, Yahoo's YUI Compressor, Closure has shown better compression rates (as seen in [5] and [6].)

It is also worth mentioning, as seen in Figure 5, the use of HTML5 specification's `history.pushState()` method, which allows to add entries to the history object of the browser. This way the user can navigate back and forth between the application's modules using the browser's Back and Forward buttons even though these modules have been loaded dynamically using AJAX calls.

6.2 Migrating from Flex to HTML5. Server Side

Internally, Flex based applications generally use the binary AMF format to communicate with the server. While there is one implementation of an AMF manager in JavaScript ([2]) it is incomplete and less efficient than not using AMF at all (the point of using AMF is to increase the communication performance). Hence, we decided to call the web services offered by the Babelium RPC API directly instead.

In order to replicate the main characteristics of a Flash based RIA, namely performance, responsiveness, and interactivity, it is paramount to allow (when possible) the use of asynchronous calls to render the presentation layer. To do that, first we have modularized our application in modules and widgets. Each section of Babelium (Practice, Evaluation, Subtitiling) forms a module. And each module is built by the combination of some widgets. By instance, the Evaluation module has widgets for showing responses of other users that have not been evaluated yet, responses the current user already evaluated and responses evaluated to the current user. This way, our application will first ask the server

```
public class ExerciseEvent extends CairngormEvent {
[...]
public function ExerciseEvent(type:String,
exercise:ExerciseVO = null, report:ExerciseReportVO =
null, score:ExerciseScoreVO = null)
        {
            super(type);
            this.exercise=exercise;
            this.report=report;
            this.score=score;
        }

public static const
 EXERCISE_SELECTED:String="exerciseSelected";
public static const
GET_RECORDABLE_EXERCISES:String="getRecordabl
eExercises";
```

```
var ExerciseEvent = Cairngorm.Event.extend(
{

    init : function ( type, exercise, report, score )
      {
          this._super(
          type,
           {"exercise" : exercise,
            "report" : report,
            "score" : score});
    }});

// Constants
ExerciseEvent.EXERCISE_SELECTED =
"exerciseSelected";
ExerciseEvent.GET_RECORDABLE_EXERCISES
= "getRecordableExercises";
```

Fig. 4: Excerpt of Babelium's original ActionScript vs. translated Javascript (Event class)

```
public class GetRecordableExercisesCommand
    implements ICommand, IResponder
    {
public function execute(event:CairngormEvent):void
        {
            new
ExerciseDelegate(this).getRecordableExercises();
        }

 public function result(response:Object):void
        {
            // manage the response data
    }

public function fault(info:Object):void
        {
            CustomAlert.error(ResourceManager.getIn
stance().getString('myResources','ERROR_WHILE_RETR
IEVING_EXERCISES'));

        }
    }
```

```
var GetRecordableExercisesCommand =
   Cairngorm.Command.extend(
  {
    execute : function ()
    { // ...
        var _this = this;
      BP.PracticeDelegate
                .getRecordableExercises(_this);
    },
    onResult : function ( response )
    {
   history.pushState("Exercise", {module :
                "practice" }, "?module=practice");
   BP.CMS.manage(response);
    },

    onFault : function ()
    {
        BP.CMS.abortLoading();
        alert("{{$ERROR_LOADING_EXERCISE
_MODULE}}");
    }});
```

Fig. 5: Excerpt of Babelium's original ActionScript vs. translated Javascript (Command class)

to get the whole module (evaluation module) and then, whenever the user clicks on a specific tab ("Responses awaiting evaluation"), an event will trigger an asynchronous request to get the specific widget, and only that, without reloading the page. Following the same example, the list of specific responses will be asked using Babelium's RPC API. Figure 6 depicts this explanation also showing two main entry-points to the server infrastructure, one for retrieving full modules and specific widgets along with their data (bridge.php) and another one for retrieving data (api/rest.php). There is one more issue related to where do we

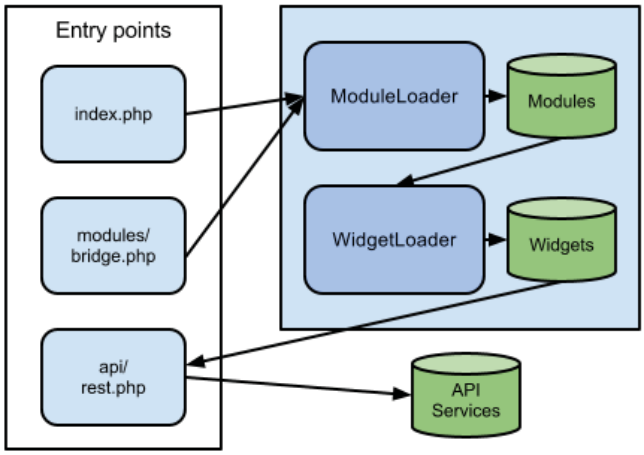


Fig. 6: Modules and widgets are served through the use of a REST API

save and manage the HTML layout in the server. This is done through the use of Smarty ([14]), a library that allows the creation of HTML templates that are imported and used in PHP scripts to ease the process of web page design. While allowing the presentation of a web page to change separately from the back-end, Smarty is also useful for caching pages, thus allowing to increase the web server's performance.

Finally, Figure 7 summarizes the main blocks of our proposal to translate a Cairngorm based RIA to an almost fully HTML5 compliant application.

7 The need for Flash widgets

As of today, a full translation to HTML5 of a multimedia application such as BabeliumProject is not possible because there are some shortcomings related to HTML5 device access APIs. Indeed, even though the W3C specification defines a way to access both webcam and microphone devices using Javascript ([19], [17]),

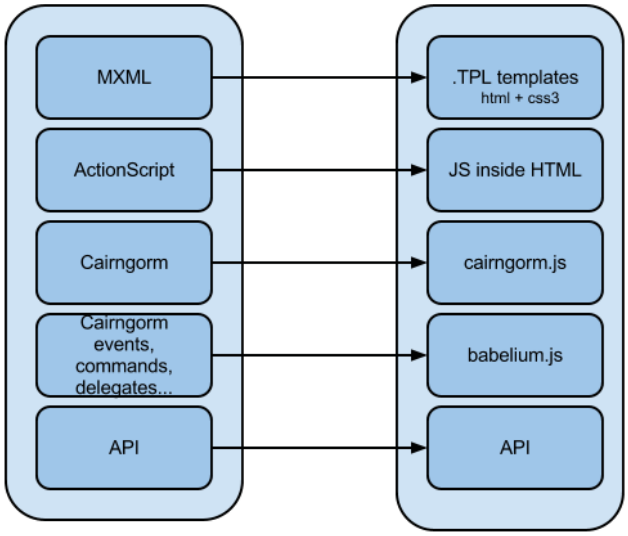


Fig. 7: Mapping of a Cairngorm based Flex RIA's blocks to HTML5

and beta versions of both Chrome and Opera have implemented that draft to a certain degree, it is not fully implemented in the stable version of any browser. In these cases, developers must seek a compromise between what is wanted (a full HTML5 version of a multimedia RIA) and what is available (only Flash/Flex and other plugin-based solutions offer webcam and audio support).

In order to maximize the decouple of the inevitable use of a plugin-based widget (in our case, a Flex based widget) from the rest of the HTML5 application, we propose to wrap the Flex widget using an HTML `<object>` tag and to communicate each other using the ExternalInterface API.

7.1 HTML5 and Flex widgets integration

In Flash 8, Adobe included an official API to enable integration between a Flash application and the browser called the "ExternalInterface API". The API consists of two main functions: the `addCallback()` function that exposes an ActionScript function to the Javascript code, and the `call()` function to invoke JavaScript functions from ActionScript. BabeliumProject has used both functions to wrap a Flex widget inside an HTML5 page. This way, the use of a plugin-based widget is constrained to a minimal SWF file that allows to capture webcam's and microphone's signals. Should the WebRTC initiative finalize its implementation, it will be an easy task to replace it by a purely HTML widget.

8 Conclusions

HTML5 has proved to be a real alternative to RIA development. The HTML5 specification is still a work in progress but there are already a lot of features implemented by major browsers, not only in the desktop but also in the mobile arena (smartphones, tablets, netbooks). Even a pioneering company in the RIA market like Adobe has acknowledged that the future of RIA development is tied to HTML5, signaling web engineers and developers the way to go. Today there are a lot of web applications requiring a plugin to work properly (namely Flash, Silverlight or JavaFX plugins). These plugins are in many cases unavailable for some smartphone operating systems or consume too much resources (specially CPU) for the hardware of these phones. So there is a real need to migrate some RIA applications from plugin-required solutions to HTML5 applications. In this paper we analyze a real case of migrating a Flex application that uses an architectural framework called Cairngorm. Flex is in turn an application framework for ActionScript developers.

We have shown that there are already some third party applications with the aim of helping the developer in the automatic translation from ActionScript to Javascript. In the future, it would be interesting to use these applications to produce Javascript translations of the same source code (Cairngorm classes) and compare the results between each other and with our tailor-made solution, but as of today none of the solutions take advantage of the fact that Flex applications might be written using an architectural framework. In those cases, we advocate for the idea that it will be very interesting and useful to have a similar framework already translated to HTML5, easing the process of the migration for the developers. As an example, we have shown a JavaScript implementation of the Cairngorm framework that has allowed us to migrate the Babelium application following its original architecture in a fast way.

There have been some issues in the migration of the code related to webcam and microphone access due to the fact that support for HTML5 is yet a work in progress. On those cases we advocate for wrapping a minimal Flash widget that implements currently unsupported HTML5 features and communicates with the rest of the application through the use of a Javascript API. This way it will be easy to replace the Flash widget when, in the near future, major browsers implement those features.

References

1. Adobe's Blog, http://blogs.adobe.com/flex/2011/11/your-questions-about-flex.html
2. Amf.js, a pure Javascript AMF implementation, James Ward, 2010, http://www.jamesward.com/2010/07/07/amf-js-a-pure-javascript-amf-implementation/
3. Cairngorm, architectural framework for Flex RIAs, http://opensource.adobe.com/wiki/display/cairngorm/UnderstandingCairngorm, 2010

4. Closure, a Javascript Compiler, https://developers.google.com/closure/compiler/
5. Google Closure Compiler vs. YUI Compressor, http://www.bloggingdeveloper.com/post/Closure-Compiler-vs-YUI-Compressor-Comparing-the-Javascript-Compression-Tools.aspx, 2009
6. Google Closure vs. YUI Compressor, http://blog.feedly.com/2009/11/06/google-closure-vs-yui-min/, 2009
7. FalconJS, blog entry by Paradise, B., http://blogs.adobe.com/bparadie/2011/11/19/what-is-falconjs/, 2011
8. Fraternali, P., Rossi, G., Sánchez-Figueroa, F., Rich Internet Applications IEEE Internet Computing, 9-12, V. 14, 2010
9. Gordon, https://github.com/tobeytailor/gordon
10. Jangaroo, http://plastic.host.adobe.com/plastic4.pdf
11. Meliá, S et al., OOH4RIA Tool: Una Herramienta basada en el Desarrollo Dirigido por Modelos para las RIAs. JISBD, 219-222, 2009
12. Opdyke, W., Refactoring Object-Oriented Frameworks. Ph.D.Thesis, University of Illinois at Urbana-Champaign, 1992.
13. Preciado, J.C. et al., Designing Rich Internet Applications Combining UWE and RUX-Method. ICWE, 148-154, 2008
14. Smarty, a PHP template engine, http://www.smarty.net/
15. StatOwl, a comparison between Silverlight, Flash and JavaFX market penetration, http://www.statowl.com/custom_ria_market_penetration.php
16. Swiffy, http://www.google.com/doubleclick/studio/swiffy/
17. W3C's HTML5 Specification for getting access to local devices, http://dev.w3.org/2011/webrtc/editor/getusermedia.html
18. Wallaby, http://labs.adobe.com/wiki/index.php/Wallaby
19. WebRTC's blog page, http://www.webrtc.org/
20. Wischusen, J., Professional Cairngorm. Wiley, Indianapolis, 2010

Verificación de la información extraída por wrappers web usando algoritmos basados en colonias de hormigas

Iñaki Fernández de Viana, Pedro J. Abad, José Luis Álvarez, José Luis Arjona

Dpto. Tecnologías de la Información, Universidad de Huelva.
21071 La Rábida (Huelva)
{i.fviana, abadhe, jose.arjona, alvarez}@dti.uhu.es

Resumen Un *wrapper* es un sistema automático que permite navegar, extraer, estructurar y verificar información proveniente de la Web. Una de las tareas más importantes dentro de este campo es la verificación automática de la información proveniente de esta fuente de datos semi-estructurados. En la literatura existen distintas técnicas que intentan solucionar este problema. En este trabajo, presentamos una nueva propuesta que hace uso de los algoritmos de optimización basados en colonias de hormigas. De los distintos algoritmos de colonias de hormigas existentes, usaremos el denominado *Best-Worst Ant System* que ya ha sido usado en diversos problemas de optimización alcanzando unos resultados bastante prometedores. Realizaremos un análisis no paramétrico del comportamiento de nuestra propuesta y la compararemos con las técnicas de verificación ya existentes. Para hacer este estudio utilizaremos diversas bases de datos reales. Los resultados obtenidos nos permiten confirmar el buen rendimiento que presenta nuestra propuesta frente a los métodos tradicionales aplicados.

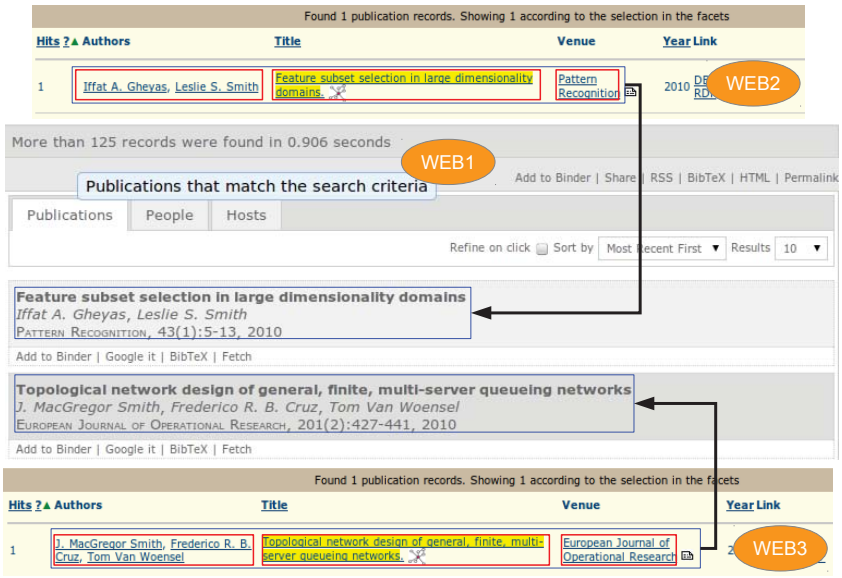
1. Introducción

En los últimos años la Web se ha convertido en la principal fuente de información, millones de islas amigables de datos proporcionan información mediante interfaces vistosas y de fácil uso por parte de los usuarios. En cambio, estas islas son difíciles de integrar dentro de un proceso automático de negocio ya que raramente proporcionan una interfaz de programación que nos permita obtener una visión estructurada de la información que éstas contienen.

Si queremos automatizar este proceso de integración tenemos que usar *wrappers*. Un *wrapper* es un sistema que proporciona una interfaz de programación para estas islas emulando la interacción que un usuario tendría con ellas. En [9] se muestra una visión global del marco de trabajo para el desarrollo de *wrappers* basado en la literatura especializada. Este marco de trabajo se divide en tres grandes fases: gestor de formularios, navegador y extractor de información. Desgraciadamente, ninguno de estos elementos está al 100 % libre de errores ya que la Web es, por definición, un ente dinámico que sufre habitualmente cambios en sus formularios de búsqueda o en sus diseños.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 185-198, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

Figura 1. Servicio de metabúsqueda bibliográfica



Para introducir el problema al que nos enfrentamos y justificar la necesidad de tratarlo, consideraremos la web que se muestra en la parte central de la Figura 1. *WEB1* es un metabuscador de referencias bibliográficas que aporta una serie de valores añadidos a los servicios ya existentes en este campo. Cuando usamos su formulario de búsqueda (en este caso nos interesamos por autores cuyo nombre contenga la cadena "Smith"), *WEB1* hace uso de otros portales de referencias bibliográficas, *WEB2* y *WEB3*, para obtener los datos que el usuario espera ver. Para lograr esto, es necesario que *WEB1* sea capaz de extraer los datos de interés de *WEB2* y *WEB3*. A tenor de lo que se muestra en la Figura 1, parece claro que *WEB1* puede obtener esta información de *WEB2* y *WEB3*, si extrae los nombres de los autores de la primera columna, los títulos de los artículos de la segunda y el nombre de la conferencia de la tercera. Así pues, parece sencillo generar un mecanismo automático que haga esta extracción. Ahora bien, imagine que los diseñadores de *WEB2* y *WEB3* deciden cambiar el orden de presentación del título del artículo y del nombre de la revista pasando a ocupar las columnas tres y dos respectivamente. Las implicaciones que este cambio tendría en *WEB1* son que mostraría como título del artículo el nombre de la revista y como nombre de la revista el nombre del artículo. Estos problemas se podrían repetir ya que los diseñadores de *WEB2* y *WEB3* pueden realizar cuantos cambios consideren sin avisar a los responsables de *WEB1*.

Para evitar este tipo de problemas, los *wrappers* incluyen un sistema de verificación cuyo objetivo es detectar incorrecciones en los datos obtenidos por

el extractor de información. Al problema de verificar la información extraída por un *wrapper* lo denominaremos *Wrapper Verification Problem* (WVP)[16]. Como se desprende del ejemplo anterior, si los *wrappers* adolecen de este elemento, las diferentes aplicaciones que los usan estarían trabajando con información, posiblemente, inconsistente.

El uso de *wrappers* web es una práctica muy común [3,14,17,22,25,27] en áreas como la inteligencia competitiva, la comparación de tiendas o integración B2B. La dependencia, cada vez mayor, de los datos disponibles en la Web ha ocasionado que muchos investigadores profundicen en el estudio del WVP [18,20,25,6,7,16]. La técnica que aquí se propone pretende solucionar las carencias detectadas [8] en las técnicas tradicionalmente usadas en el campo de la verificación de información de extractos web [16,18,20] relacionadas con la heterogeneidad de los datos, la dependencia de un modelo predeterminado de datos, la dependencia de las características usadas para generar dichos modelos o las distintas funciones usadas para combinar la información suministrada por cada una estas características.

En este trabajo proponemos el uso de un nuevo método basado en algoritmos de colonias de hormigas (ACO) como ejemplo de cómo aplicar técnicas usadas en la resolución de problemas de optimización combinacional al WVP. ACO [13,11] es uno de los paradigmas de algoritmos bioinspirados más extendidos. Ha demostrado un buen comportamiento a la hora de resolver problemas de optimización combinacional [4]. La principal ventaja que presenta enfocar el WVP como un problema de optimización combinacional es que abrimos la posibilidad de aplicar un gran número de técnicas a un problema en el que tradicionalmente se han venido aplicando un número muy reducido. Además de reformular el WVP desde el punto de vista de la optimización combinacional, hemos definido elementos propios de las metodologías ACO como son: la información heurística, la inicialización de la feromona, la construcción de la solución, la función de coste y la actualización de la feromona.

Veremos que las técnicas basadas en ACO son muy adecuadas para resolver el WVP alcanzando unos resultados superiores a las métodos actualmente usados [18,20,16]. De los diferentes algoritmos ACO existentes, aplicaremos el denominado *Best-Worst Ant System* (BWAS) ya que ha sido aplicado para resolver diversos problemas combinatorios demostrando un comportamiento muy prometedor [5,21,15]. Ahora bien, la aplicación de cualquier otro algoritmo ACO sería inmediata con los datos que en este trabajo se aportan.

Este artículo se ha estructurado de la siguiente manera. En la Sección 2 describiremos el WVP. En la Sección 3 introduciremos los conceptos básicos de la metaheurística ACO y del algoritmo BWAS. En la Sección 4 veremos cómo aplicar las metodologías ACO al WVP. En la Sección 5 detallaremos el diseño de la experimentación y discutiremos sobre los resultados alcanzados. Finalmente, en la Sección 6 indicaremos una serie de conclusiones y trabajos futuros.

2. Verificación de información

Los extractores de información están formados por una serie de reglas de extracción inferidas de un conjunto de datos. Cuando estas reglas se basan en etiquetas *HTML*, estos extractores sólo puede usarse en las fuentes de información de las cuales provienen los datos usados para inferir dichas reglas. Esto es, si la fuente cambia, en algunos casos, el extractor puede devolver resultados no deseados [16]. A menos que la información extraída por el *wrapper* sea verificada de una forma automática, estos datos pasarían inadvertidos para las aplicaciones que los usan.

El WVP, como parte integradora de un *wrapper,* ha sido propuesto por diferentes autores desde diferentes puntos de vista [18,20,6,25,16]. En [9] se propuso un marco de trabajo que pretende homogeneizar todos los conceptos y técnicas existentes en el campo de la verificación. A modo de resumen y valiéndonos del ejemplo propuesto en la Figura 1, el proceso de construcción de un verificador empieza con la recopilación de un conjunto de datos $S = \{s_1, s_2, \ldots s_M\}$, donde cada s_i es un *slot* y el término *slot* se refiere tanto a un atributo como a un registro; M es el número de *slots*. Un atributo a es una cadena contenida en una página web referenciable con algún tipo de localizador. Un registro es un conjunto de atributos. Cada *slot* $s_i = (a_i, c_i) \mid c_i \epsilon [1, N],\ i = 1..M$ es etiquetado con una clase c que denota su rol existiendo un total de N clases.

Para nuestro ejemplo, los atributos serían las cadenas: *("Feature subset selection in large dimensionality domains", "Iffat A. Gheyas", "Leslie S. Smith", "Pattern Recognition", "Topological network design of general, finite, multi-server queueing networks", "J. MacGregor Smith", "Frederico R. B. Cruz", "Tom Van Woensel", "European Journal of Operational Research")*; existirían tres clases: título, autor y revista. Así, el conjunto S estaría formado por *S={("Feature subset selection in large dimensionality domains", título), ("Iffat A. Gheyas", autor), ("Leslie S. Smith", autor), ("Pattern Recognition", revista), ("Topological network design of general, finite, multi-server queueing networks", título), ("J. MacGregor Smith", autor), ("Frederico R. B. Cruz", autor), ("Tom Van Woensel",autor), ("European Journal of Operational Research", revista)}*.

A partir de S generamos el modelo de verificación V_S que es una modelización de S basada en la aplicación de una serie de características. V_S se define como $V_S = \{(x_1, c_1), (x_2, c_2), \ldots (x_M, c_M)\}$ donde $x_i = [x_{i1}, x_{i2}, \ldots, x_{id}] = [f_1(a_i),\ f_2(a_i), \ldots, f_d(a_i)] \mid a_i \epsilon S$ y d es el número de características y establece la dimensionalidad del problema. Una característica f es una descripción cuantificable de un *slot*. A modo de ejemplo, podemos definir f_1 como aquella característica que cuenta el número de palabras que contiene una cadena y como f_2 la que indica si una cadena contiene símbolos de puntuación. Al aplicarlas, el modelo de verificación que obtendríamos sería: *V_s= {([7,0], título), ([3,1], autor), ([3,1], autor), ([2,0], revista), ([9, 1], título), ([3, 1], autor), ([4, 1], autor), ([3, 0],autor), ([5, 0], revista)}*.

Una vez construido V_S es necesario inferir una función *m(x)* de tal forma que para un vector x dado, *m(x)* devuelve una estimación de lo similar que es x

respecto a V_S:

$$\begin{array}{rccc} m: & \mathbb{R}^d & \longrightarrow & \mathbb{R} \\ & [x_{i,1} \ldots x_{i,d}] & \longmapsto & \{0,1\} \end{array}$$

Después de inferir esta función, tendremos creado el verificador. Las funciones más conocidas son las propuestas en [18,20,25,16]. Resumidamente indicaremos que en [16] los datos se modelan mediante distribuciones normales, en [18] se usa el valor medio, en [20] se obtiene un modelo parecido al indicado en [16] pero se modifica la forma en la que se estima la probabilidad de que un valor siga una normal. Por último, en [25] se mejora la característica *DataProg* usada en [18]. Son, por tanto, técnicas puramente estadísticas que no incorporan ningún mecanismo de búsqueda.

Llegados a este punto, el *wrapper* estará listo para ponerse en producción y comenzar a obtener información de los sitios web para los que fue diseñado. A partir de ahora, el *wrapper* extraerá una serie de datos no verificados U. En nuestro ejemplo, el *wrapper* puede extraer datos relacionados con autores que tengan, por ejemplo, la cadena "John". El resultado será un conjunto de datos del tipo *U={("A System to Detect Inconsistencies between a Domain Expert's Different Perspectives on (Classification) Tasks", título), ("Derek H. Sleeman", autor), ("Andy Aiken", autor), ("Laura Moss", autor), ("John Kinsella", autor), ("Malcolm Sim", título), ("Advances in Machine Learning II", revista)}*. La diferencia fundamental entre S y U es que el primero está correctamente etiquetado (el proceso ha sido supervisado por un experto) mientras que el responsable de verificar que las etiquetas en U son correctas es $m(x)$. En nuestro ejemplo, si *m(x)* está correctamente definida, debería devolver un 1 para todos los *slots* salvo para el *slot ("Malcolm Sim", título)* que está etiquetado como un título cuando no lo es. En este último caso lanzaría una alarma para que un experto determine si el *slot* está o no mal etiquetado.

3. Optimización mediante algoritmos de colonias de hormigas

Los algoritmos ACO [13] se engloban dentro de los métodos de búsqueda bio-inspirados y fueron propuestos por primera vez en [12]. A partir de este modelo, se han ido desarrollando diversos algoritmos que han permitido solucionar multitud de problemas de optimización combinacional [10,11,4].

El funcionamiento de los algoritmos ACO se basa en emular el comportamiento social que una colonia de hormigas tiene cuando busca comida. Durante este proceso de búsqueda, las hormigas reiteradamente salen y vuelven al nido depositando una sustancia denominada feromona. Esta sustancia es la que guiará el proceso de búsqueda. Cuando una hormiga sale del nido en busca de comida, toma la decisión de ir hacia una dirección u otra atendiendo a la cantidad de feromona que encuentre en cada una de las posibles direcciones. Por tanto, el resultado final de este proceso será la creación de un camino con altas cantidades de feromona que une el nido con la comida y que será utilizado por la totalidad de la colonia. La feromona es un componente químico que sufre evaporación. Así

Algoritmo 1 Algoritmo BWAS

1. Asignamos un valor inicial de feromona a todos los arcos
2. Mientras (no se cumpla la condición de finalización)
   - *a*) Generamos un camino para cada hormiga
   - *b*) Aplicamos el mecanismo de evaporación de la feromona
   - *c*) Aplicamos un proceso de búsqueda local para mejorar la mejor solución actual
   - *d*) Actualizamos los valores de la mejor y peor hormiga
   - *e*) Aplicamos la regla de actualización de la feromona propia del algoritmo BWAS
   - *f*) Aplicamos mutaciones en la cantidad de feromona de algunos arcos
   - *g*) Si el algoritmo se queda estancado en un máximo local reiniciamos

pues, los caminos que perdurarán serán aquellos más cortos y que, por tanto, sufren menos evaporación ya que las hormigas tardan menos en pasar por ellos.

El funcionamiento básico de un algoritmo ACO se puede describir de la forma siguiente [12]: para cada iteración, una colonia de hormigas se va desplazando por un grafo que representa una instancia del problema atendiendo a una regla de transición probabilística que depende de la información heurística y la información memorística (esta última viene dada por la cantidad de feromona depositada). Una vez que cada una de las hormigas de la colonia ha acabado de desplazarse por el grafo se actualiza la cantidad de feromona depositada en él. La regla de actualización global de la feromona decrementa uniformemente la cantidad de feromona (simula la evaporación) e incrementa el valor de feromona en aquellas aristas del grafo por las que han pasado las hormigas atendiendo a la calidad de las soluciones alcanzadas por cada una de ellas.

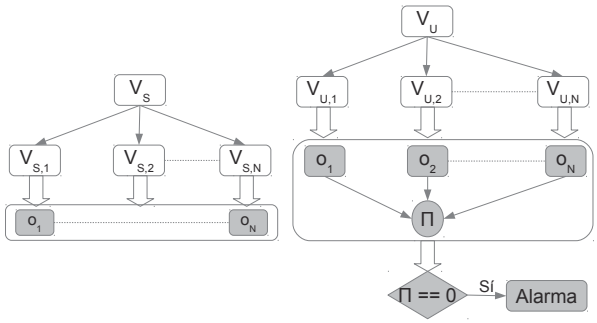
Si bien en la literatura podemos encontrar multitud de variantes del algoritmo ACO inicial [19,23,2], en este trabajo usaremos el algoritmo BWAS [21] que se caracteriza por intentar mejorar el rendimiento del ACO original usando conceptos de algoritmos evolutivos. Así, la regla de actualización global de la feromona se basa en lo propuesto en el algoritmo *Population-Based Incremental Learning* [1](tiene en cuenta las mejores y las peores soluciones) además de introducir un operador de mutación y reinicialización de la cantidad de feromona. Estos mecanismos son introducidos por los autores para evitar el estancamiento en el proceso de búsqueda. El Algoritmo 1 describe resumidamente el funcionamiento del BWAS.

Para poder aplicar el algoritmo BWAS a un problema de optimización combinacional es necesario definir los siguientes aspectos [4]: la representación del problema, la información heurística, la inicialización de la feromona y la función objetivo.

4. Aplicando técnicas ACO al problema de la verificación

Como se indica en la Figura 2.a, antes de aplicar la metaheurística ACO al WVP, es necesario que el modelo de verificación V_S se divida en N subconjuntos, uno por cada una de las clases presentes en el conjunto S [6]. Cada uno de

Figura 2. Construcción y uso del verificador



(a) Construcción del verificador C(x) a partir de S (b) Verificación de un conjunto U

estos subconjuntos están compuestos por ejemplos de la misma clase: $V_{S,k} = \{(x,c)\epsilon V_T \mid c = k\}\ ,\ k = 1 \ldots N$.

A cada uno de estos N subconjuntos le aplicaremos un algoritmo ACO para inferir la función *m(x)* discutida en la Sección 2. Al tener N clases distintas *m(x)* pasaría a estar formada por una serie de funciones $C(x) = \{o_1, o_2, \ldots o_N\}$ donde o_i es una función que se obtendrá a partir de técnicas ACO:

$$\begin{array}{rccc} o_i: & \mathbb{R}^d & \longrightarrow & \mathbb{R} \\ & [x_{i,1} \ldots x_{i,d}] & \longmapsto & \{0,1\} \end{array}$$

Esto es, C es una colección de N funciones donde o_i ha sido inferida usando *slots* de la clase *i*.

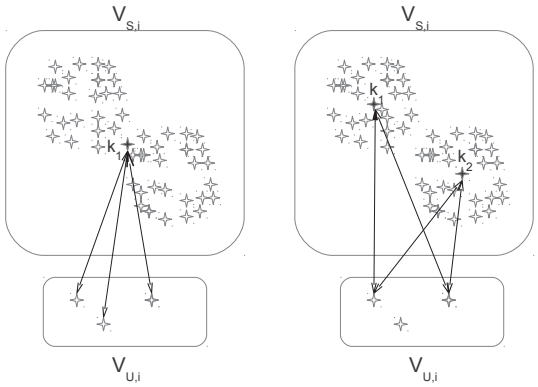
Una vez que tenemos construido C, podemos verificar cualquier conjunto U (Figura 2.b) aplicando $Y = \prod_{i=1}^{N} O_i(x) \quad \forall\ (x,c)\epsilon V_U$ donde $O_i(x) = o_k(x)\ /\ (x,c)\epsilon V_{U,k}$. Por tanto, para cada uno de los *slots* que forman U consultamos su clase y aplicamos la función o_i oportuna. Así pues,, para todos los *slots* etiquetados con la clase c_1 aplicamos la función o_1, para los etiquetados como c_2 la o_2 y así sucesivamente. Si $Y = 0$ entonces alguno de los *slots* ha sido mal etiquetado y, por tanto, U es incorrecto y debe ser revisado por un experto.

En los siguientes apartados analizaremos cómo hemos construido las funciones o_i basándonos en el algoritmo BWAS.

4.1. Hipótesis inicial

El objetivo de las funciones o_i es devolver un 1 si la clase del atributo es correcta o 0 si no lo es. Por tanto, podríamos considerar que cada una de estas funciones representa un problema de asignación de tareas tipo QAP (*Quadratic Assignment Problem*) [5]. Las tareas son los *slots* pero no tendríamos definidas las localizaciones.

Figura 3. Representación de $V_{S,k}$ mediante un único punto o varios puntos



(a) Representación 1-punto (b) Representación 2-punto

Estas localizaciones podrían ser puntos establecidos aleatoriamente dentro del espacio d-dimensional en el que se posicionan los *slots*. Esta idea no es nueva, en [18] el conjunto V_S se representa mediante el punto medio (Figura 3.a). Ahora bien, como el conjunto V_S es heterogéneo [8], sería deseable encontrar más de un punto que represente a V_S (Figura 3.b). Por tanto, lo que pretendemos hacer con el algoritmo BWAS es buscar una serie de K-puntos $k = \{k_1, k_2, ...k_K\}$ $k_i \in \mathbb{R}^d$ que describan adecuadamente a V_S. Así pues, el verificador estaría compuesto por los K puntos localizados tras la aplicación del BWAS al conjunto V_S. Cuando el *wrapper* pase a producción, la función o_i calcularía para cada uno de los elementos de U las distancias a cada uno de los elementos de k. Si la distancia mínima es más pequeña que un umbral pre-establecido entonces el nuevo elemento es similar a lo contenido en V_S y, por tanto, devolverá un 1 (vea la parte inferior de la Figura 3).

4.2. Representación del problema para el WVP

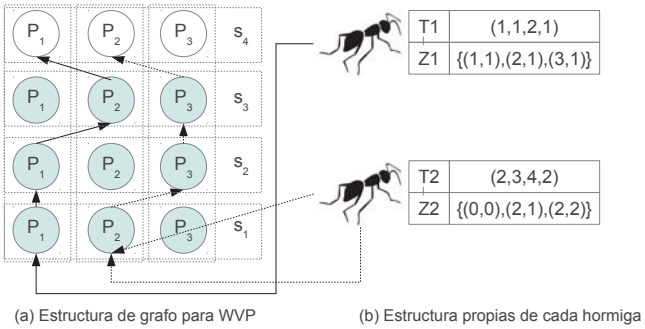
En los algoritmos ACO, el espacio de búsqueda se representa como un grafo etiquetado por el que las hormigas se van moviendo.

En nuestro caso particular, este grafo se podría representar mediante una matriz de tamaño $M \times K$. Cada arista *A(i,j)* representa el hecho de asociar el *slot* i al punto j. Cada *slot* i sólo podrá ser asignado a un punto j.

En la Figura 4 se muestra cómo sería este grafo. En él se representa un problema ficticio de verificación en el que V_S está formado por cuatro *slots* cada uno de los cuales ha sido modelado por dos características (d=2). V_S debe ser representado mediante tres puntos usando para ello dos hormigas. En esta figura se representa con un trazo continuo el rastro dejado por la primera hormiga y con uno punteado el que ha dejado la segunda.

Cada hormiga i tiene asociada una matriz T_i donde va almacenando las decisiones tomadas conforme ha ido asociando *slots*. Por ejemplo, la primera hormiga

Figura 4. Representación de WVP como grafo etiquetado



tiene almacenado $T_1 = (1, 1, 2, 1)$ que se debe interpretar cómo: el primer y segundo *slot* han sido asignados al primer punto, el tercer *slot* al segundo punto y el cuarto *slot* al primer punto. Además, cada hormiga i tiene una matriz Z_i de dimensiones $K \times d$ donde almacena las coordenadas de los puntos calculados.

4.3. Algoritmo BWAS aplicado al WVP

Una vez descrita la representación que hemos usado, detallamos los pasos que hemos seguido a la hora de aplicar el BWAS.

1. Inicializamos la matriz de feromona a un valor fijo muy bajo: $\tau_o = 0{,}001$
2. Para cada una de las hormigas, inicializamos el valor de Z haciendo coincidir cada punto k_i con alguno de los *slots* de V_S.
3. Para cada una de las iteraciones
   a) Para cada hormiga i contenida en la colonia
      1) Inicializamos el valor de $T_i = \varnothing$
      2) Para cada *slot* s contenido en V_s
         a' Seleccionamos aleatoriamente un *slot* s no contenido en T_i
         b' Asignamos el *slot* s a un punto k aplicando la fórmula de construcción de las soluciones descrita en [5] teniendo en cuenta que la información heurística asociada a la decisión por parte de la hormiga i de asignar s al punto k viene dada por $\eta^i_{s,k} = 1/d(i, s, k)$ donde $d(i, s, k) = \sqrt{\sum_{v=1}^{d}(x_{s,v} - Z_{i,k,v})^2}$.
   b) Para cada hormiga i contenida en la colonia
      1) Evaluamos la solución alcanzada aplicando la fórmula siguiente: $fit(i) = \sum_{i=1}^{M}\sum_{j=1}^{K} w_{ij}\sqrt{\sum_{v=1}^{d}(x_{i,v} - Z_{i,k,v})^2}$ donde w_{ij}= 1 si el slot i ha sido asignado al punto j y $w_{ij} = 0$ en otro caso.
      2) Para cada uno de los K puntos que representan a V_S actualizamos sus coordenadas aplicando la fórmula: $Z_{i,k} = \frac{\sum_{i=1}^{M} w_{ik}s_i}{\sum_{i=1}^{M} w_{ik}}$
   c) Evaporamos, añadimos, mutamos la feromona y reiniciamos el proceso de búsqueda (si fuera necesario) tal y como se propone en el algoritmo original BWAS [21].

5. Experimentación y resultados

En esta sección se describe cómo se ha diseñado la experimentación [26] para comparar el rendimiento de los distintos algoritmos estudiados en este trabajo.

5.1. Descripción de la base de datos y métodos comparados

Para hacer el estudio experimental, hemos usado la base de datos ya etiquetada propuesta en [16]. Está formada por 27 sitios web de distintos dominios y diversos tamaños; cada uno de los cuales tiene un número de clases que oscila entre las 2 y 8. Para nuestra experimentación, aplicaremos tantos algoritmos BWAS como clases contenidas en cada sitio web, en total 102 problemas de optimización combinatoria. Para crear los modelos de verificación hemos usado las 39 características descritas en [6,7].

Esta base de datos la usamos para comparar nuestra propuesta basada en el algoritmo BWAS con los algoritmos propuestos en [16,18,20] a los que denominaremos, respectivamente, RAPTURE, LERMAN y MCCANN.

Para los algoritmos RAPTURE, LERMAN y MACCANN el único parámetro que podemos variar es el referente al porcentaje de *target* rechazados [24] que lo hemos establecido en 0,05. Respeto al algoritmo BWAS, los parámetros usados han sido: número total de hormigas 5, número de iteraciones 100, $\tau_o = [0{,}7, 0{,}8]$, $\beta = 2$, $\alpha = 1$, $\rho = 0{,}1$. El resto de parámetros del algoritmo son los mismos a los descritos en [5].

5.2. Medidas de rendimiento y comparativas

El WVP es un problema claramente desbalanceado [7] por lo que necesitamos una medida que no se vea afectada por la distribución de clases. Por este motivo hemos usado la medida AUC (Área bajo la curva ROC).

Debido a que el test de Lilliefors nos indica que la serie de valores de AUC obtenidos en este experimento no siguen una distribución normal, hemos aplicado test no paramétricos [12, 13] para asegurar la significación estadística entre los AUC medios. Para hacer la comparación por pares hemos usado el test de Wilconxon Signed-Ranks [20], estableciendo el nivel de confianza a 0,05. Para realizar esta comparación múltiple usaremos el test de Friedman Aligned-Ranks [21] y el método post-hoc de Holm [22] con un nivel de confianza de 0,05. El test de Friedman se usa para detectar diferencias significativas entre las medidas de rendimiento de los algoritmos estudiados. Cuando esta diferencia exista, aplicaremos el método post-hoc de Holm para encontrar aquellos algoritmos que son significativamente diferentes [23].

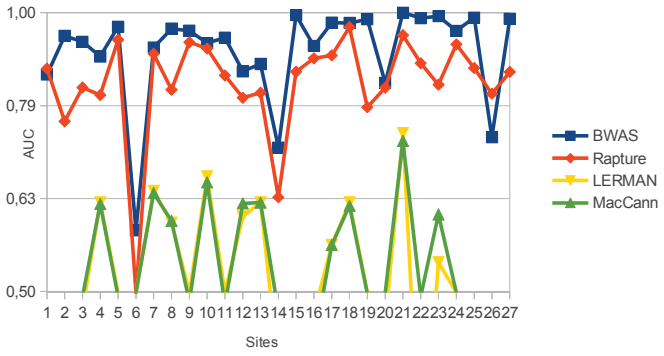
5.3. Diseño del experimento y resultados

El diseño del experimento se ha realizado de la siguiente manera. Primero cada uno de los sitios web se dividió en tantos conjuntos como clases tenga (en

total 102). Para cada una de estas particiones aplicamos un procedimiento de 10-*fold* validación cruzada usando para entrenar parte de las instancias de la clase de interés y como conjunto de test aquellas instancias de la clase objetivo no utilizadas en el test y las instancias del resto de clases. En cada una de las ejecuciones de este proceso de validación, creamos un nuevo modelo de verificación y construimos un nuevo verificador. Así pues, el total de verificadores que han formado nuestro experimento es de 4080.

En la tabla 1 se muestran los resultados medios obtenidos por las diferentes algoritmos estudiados. El detalle de este valor para cada uno de los sitios web aparece en la imagen superior. Esta media se ha obtenido después de realizar el procedimiento de validación cruzada antes indicado. En la parte inferior de la Tabla 1 aparece el resumen del valor medio de AUC de los distintos algoritmos después de haber sido aplicados a los sitios web objeto de estudio (entre paréntesis indicamos la desviación típica).

Cuadro 1. Resultados de AUC medio



BWAS	RAPTURE	LERMAN	MCCANN
0,92 (0,13)	0,85 (0,13)	0,54 (0,18)	0,55 (0,17)

En la tabla 2.a se muestra los resultados del test de Wilcoxon. En cada celda se muestra el resultado de la comparación por pares entre el algoritmo que ocupa la columna i y el que ocupa la fila j. El símbolo $\bullet$ indica que el método que hay en la fila mejora al de la columna mientras que el símbolo $\circ$ indica justo lo contrario. La significación es de 0.9 para la diagonal superior y de 0.95 para la inferior. Si la celda está vacía ninguno de los algoritmos es mejor o peor que el otro.

Por último, en la tabla 2.b mostramos los resultados logrados por el test de Friendman. En la tercera columna se muestra el valor del ranking y en la cuarta los p-valores obtenidos por Holm. Como el algoritmo BWAS es el que mejor ranking obtiene, se utiliza como referencia para calcular los p-valores.

Cuadro 2. Resultados del test de Wilcoxon y Friendman

	(1)	(2)	(3)	(4)
BWAS (1)	-	•	•	•
RAPTURE (2)	◦	-	•	•
LERMAN (3)	◦	◦	-	
MCCANN (4)	◦	◦		-

(a) Test de Wilcoxon

i	Algoritmo	Ranking	pHolm
1	BWAS	1.4278	
2	RAPTURE	1.8918	0
3	LERMAN	3.2526	0
4	MCCANN	3.4278	0.012329

(b) Test de Friendman

5.4. Discusión

En el presente trabajo se han evaluado un total de cuatro técnicas para resolver el WVP. El rendimiento de los métodos ha sido comparado usando los test de Wilcoxon, Friendman y Holm. Los resultados presentados en las Tablas 1, 2 son bastante contundentes y vislumbran un ganador claro en la comparativa. Podemos resaltar lo siguiente:

1. La media de AUC alcanzada por el algoritmo BWAS es mejor a la obtenida por el resto de algoritmos. El que más se aproxima es RAPTURE que apenas lograr alcanzar el 0.85 por el 0.92 del BWAS. Si nos vamos al detalle de cada sitio web, vemos que el BWAS es mejor o igual que RAPTURE en todos salvo en el sitio 26. Por tanto, el comportamiento del BWAS es, en general, independiente del tamaño y del dominio de la web.
2. En la segunda fila de la tabla 2.a, se observa que el algoritmo BWAS es significativamente mejor que el resto de algoritmos estudiados ya que la primera columna siempre está marcada • y la primera columna con ◦.
3. En la tercera columna de la tabla 2.b, se advierte que el AUC logrado por BWAS, 1.4278, es el menor valor obtenido por los distintos algoritmos.. Como todos los valores de la ultima columna son más pequeños que 0.05 podemos asegurar que el algoritmo BWAS es el mejor de los estudiados con un nivel de significación del 0.95.
4. Debido a problemas de espacio, no ha sido posible incluir las tablas con los tiempos empleados por los distintos algoritmos en generar los modelos de verificación (proceso *offline*) y verificar datos (proceso *online*). El tiempo empleado en verificar los datos (proceso *online*) extraídos en un registro es despreciable para todos algoritmos. No obstante, el tiempo empleado en generar los 4080 modelos de verificación (proceso *offline*) es de 3 segundo para los algoritmos LERMAN, LERMAN y MCCANN y 72 horas para BWAS.

6. Conclusiones y trabajos futuros

En este trabajo hemos presentado una nueva propuesta basada en la metaheurísticas ACO para resolver el WVP. Para poder aplicar este tipo de algoritmos, hemos tenido que reformular el WVP como si de un problema de optimización combinacional se tratara, siguiendo el marco de trabajo de creación de verificadores web propuesto en diversos trabajos. En concreto, nuestra propuesta se basa en el algoritmo BWAS.

Para comparar nuestra propuesta con las ya existentes, partimos de una serie de base de datos reales, a los resultados obtenidos les aplicamos test estadísticos no paramétricos que demuestra el buen rendimiento logrado por nuestra técnica, que supera en valores de AUC a las técnicas usadas hasta la fecha.

Quedan aún abiertos varias líneas de investigación como el calculo dinámico de K o un estudio de los parámetros óptimos del algoritmo BWAS aplicado al WVP.

Agradecimientos

Este trabajo está financiado por la European Commission (FEDER), los proyectos de investigación nacionales y andaluces TIN2007-64119, P07-TIC-2602, P08-TIC-4100, TIN2008-04718-E, TIN2010-21744, TIN2010-09809-E, TIN2010-10811-E y TIN2010-09988-E.

Referencias

1. Shumeet Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical report, 1994.
2. Bernd Bullnheimer, Richard F. Hartl, and Christine Strauï¿œ. A new rank based version of the ant system - a computational study. *Central European Journal for Operations Research and Economics*, 7:25–38, 1997.
3. Boris Chidlovskii, Bruno Roustant, and Marc Brette. Documentum eci self-repairing wrappers: performance analysis. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, SIGMOD '06, pages 708–717, New York, NY, USA, 2006. ACM.
4. Oscar Cordon, Francisco Herrera, and Thomas Stützle. A review on the ant colony optimization metaheuristic: Basis, models and new trends. *Mathware & Soft Computing*, 9:141–175, 2002.
5. Oscar Cordón, Iñaki Fernández de Viana, and Francisco Herrera. Analysis of the best-worst ant system and its variants on the qap. In *Proceedings of the Third International Workshop on Ant Algorithms*, ANTS '02, pages 228–234, London, UK, UK, 2002. Springer-Verlag.
6. Iñaki Fernández de Viana, Pedro J. Abad, José L. Álvarez, and José Luis Arjona. Applying one class classifier techniques to reduce maintenance costs of eai. In *ICSOFT (1)*, pages 41–46, 2011.
7. Iñaki Fernández de Viana, Pedro J. Abad, José Luis Arjona, and José Luis Álvarez. Toward one class classifier techniques applied to verifier information. In *6th Iberian Conference on Information Systems and Technologies (CISTI)*, 2011.
8. Iñaki Fernández de Viana, José Luis Arjona, and José Luis Álvarez. Verificación de wrappers web: Nuevas ideas. In *13th Conference on Software Engineering and Databases*, 2010.
9. Iñaki Fernández de Viana, Inma Hernandez, Patricia Jiménez, Carlos R. Rivero, and Hassan A. Sleiman. Integrating deep-web information sources. In *8th International Conference on Practical Applications of Agents and Multiagent Systems*, 2010.

10. M. Dorigo and T. Stützle. The ant colony optimization metaheuristic: Algorithms, applications, and advances. In Fred Glover and Gary Kochenberger, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, chapter 9, pages 250–285–285. Springer, New York, 2003.
11. Marco Dorigo and Gianni Di Caro. New ideas in optimization. chapter The ant colony optimization meta-heuristic, pages 11–32. McGraw-Hill Ltd., UK, Maidenhead, UK, England, 1999.
12. Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transaction on Systemaas, Man, and Cybernetics*, 26(1):29–41, 1996.
13. Marco Dorigo and Thomas Stützle. *Ant Colony Optimization*. Bradford Company, Scituate, MA, USA, 2004.
14. Emilio Ferrara and Robert Baumgartner. Design of automatically adaptable web wrappers. *CoRR*, abs/1103.1254, 2011.
15. Jorge Casillas, Oscar Cordón, Iñaki Fernández de Viana, and Francisco Herrera. Learning cooperative linguistic fuzzy rules using the best-worst ant system algorithm: Research Articles. *Journal International Journal of Intelligent System*, 20(4):433–452, 2005.
16. Nicholas Kushmerick. Wrapper induction: Efficiency and expressiveness. *Artificial Intellligence*, 118(1-2):15–68, 2000.
17. Kristina Lerman and Craig A. Knoblock. Wrapper maintenance. In Ling Liu and M. Tamer Ozsu, editors, *Encyclopedia of Database Systems*. Springer, Leipzig, Germany, 2009.
18. Kristina Lerman, Steven N. Minton, and Craig A. Knoblock. Wrapper maintenance: A machine learning approach. *Journal of Artificial Intelligence Research*, 18:2003, 2003.
19. Marco Dorigo and LM Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Trans Evol Comput*, 1:53–66, 1997.
20. Robert McCann, Bedoor AlShebli, Quoc Le, Hoa Nguyen, Long Vu, and AnHai Doan. Mapping maintenance for data integration systems. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 1018–1029. VLDB Endowment, 2005.
21. Oscar Cordón, Iñaki Fernández de Viana, and Francisco Herrera. Analysis of the Best-Worst Ant System ant it's variants on the TSP. *Mathware Soft Computing*, pages 177–192, 2002.
22. Sunita Sarawagi. Information extraction. *Foundations and Trends in Databases*, 1(3):261–377, 2008.
23. Thomas Stützle and Holger H. Hoos. MAX-MIN Ant System, November 1999.
24. D.M.J. Tax. Ddtools, the data description toolbox for matlab, Dec 2009. version 1.7.3.
25. Charalampos E. Tsourakakis and Georgios Paliourast. Wewra: An algorithm for wrapper verification. Technical Report CMU-ML-09-100, March 2009.
26. Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishersers, 2000.
27. Jiang-Ming Yang, Rui Cai, Yida Wang, Jun Zhu, Lei Zhang, and Wei-Ying Ma. Incorporating site-level knowledge to extract structured data from web forums. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 181–190, New York, NY, USA, 2009. ACM.

Computer-Aided Relearning Activity Patterns for People with Acquired Brain Injury*

Francisco Montero, Víctor López-Jaquero, Elena Navarro, Enriqueta Sánchez

Computing Systems Department, University of Castilla-La Mancha,
Avda. España s/n Albacete 02071
{fmontero, victor, enavarro, enri}@dsi.uclm.es

Abstract. This paper describes the activities carried out in the course of developing a tool, HABITAT, to assist people with Acquired Brain Injury (ABI).

1 Context of the proposal: people with Acquired Brain Injury

People with Acquired Brain Injury (ABI) are people who have suffered "damage to the brain that occurs after birth and which is not related to congenital disorders, developmental disabilities, or processes that progressively damage the brain". There are several causes of ABI. The most frequent, present in at least 50% of cases, is cerebral vascular pathology due to degeneration of the blood vessels altering the blood flow and/or producing a brain haemorrhage. Another frequent cause is skull-brain trauma due to motor vehicle accidents, falls or physical aggression. The impact of ABI is wide-ranging. It can affect a person's social life and their development. The multitude of physical effects might include muscle spasticity, paralysis or weakness, blurred vision or decreased coordination. ABI also affects a person's cognitive abilities, such as memory, thinking skills, concentration, and organisation and planning abilities.

Relearning is critical during the first two years after the onset of damage, as most of the recovery of their previous abilities is achieved during this period. Unfortunately, several difficulties prevent them from gaining access to this process. These are mainly: (1) the difficulty of access to relearning centres due to the mobility problems of the handicapped; (2) the limited time available to perform their relearning tasks due to the small numbers of staff at the centres; (3) long waiting lists due to the increasing number of persons suffering from brain damage. E-learning can provide a means of compensating for some of the difficulties previously identified. The HABITAT (HCI techniques for ABI TreAtmenT [1]) system was developed in this context. This system was designed to exploit both e-learning and HCI approaches, so as to provide this collective with a virtual space where they can put into practice their relearning process by themselves or supervised by a specialist or a relative.

* This work has been partially supported by grant (PEII09-0054-9581) from the Junta de Comunidades de Castilla-La Mancha and also by a grant (TIN2008-06596-C02-01) from the Spanish Government.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 199-200, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

2 Relearning Activity Patterns for HABITAT: research method

During two years a continuum tracking of the relearning process involving specialists and handicapped was carried out by researchers at the University of Castilla-La Mancha in collaboration with the ABI Association of Castilla-La Mancha (ADACE-CLM). This tracking was designed to study how the process could be supported by software, that is, what features should be provided by HABITAT. With this aim, the following tasks were carried out:

- Attending relearning sessions. The main objective was to determine the nature of the interaction between the patient and specialist and how it could be improved by the use of software.
- Analysing the material used by specialists during the training activities. The result of this analysis was the development of a catalogue of *RElearning Activity Patterns* (ReAP) in HABITAT. Each ReAP has been described by: (i) *pattern name* is a handle we can use to describe a relearning activity, its solution, and consequences in a word or two; (ii) *intent* or *problem* describes when to apply the pattern and might describe specific re-education activities, such as how to improve our memory; (iii) *solution* describes the elements that make up the relearning process, their relationships, responsibilities, and collaborations; (iv) the *consequences* are the results and trade-offs of applying the pattern and the achieved individual evolution; (iv) *implementation* provides details for the development; (v) *example* provides some snapshots of the learning cards or activities used traditionally during the relearning process. Because we have identified 23 ReAPs, a proper organization strategy is required that helps in learning the pattern catalogue faster, as long as it can direct efforts to find new patterns as well. We classify ReAPs according to two criteria. The first, called *purpose*, reflects what a pattern does and improves. The identified ReAPs can treat motor, cognitive, emotional or behavioural deficits. The second criterion, called *scope*, specifies whether the pattern can be applied to the different ABI groups.
- Determining the acceptability of HABITAT ReAPs for users. A usability testing activity was carried out and a SUS-based questionnaire was compiled for ABI patients and their relatives as goal population. This acceptability testing showd that specialists and patients with ABI were receptive and motivated to introduce computers for supporting relearning activities.

References

1. F. Montero, V. López-Jaquero, E. Navarro, and E. Sánchez, "Computer-Aided Relearning Activity Patterns for People with Acquired Brain Injury," Computers & Education doi:10.1016/j.compedu.2010.12.008, 57(1):1149-1159, August 2011,

Exploring Tabletops as an Effective Tool to Foster Creativity Traits

Alejandro Catala[1], Javier Jaen[1], Betsy van Dijk[2], Sergi Jordà[3]

[1]ISSI-Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València
Camí de Vera s/n, 46022, Valencia Spain
{acatala, fjaen}@dsic.upv.es

[2]Human Media Interaction University of Twente
Faculty EEMCS
PO Box 217 7500 AE Enschede The Netherlands
e.m.a.g.vandijk@utwente.nl

[3]Music Technology Group Universitat Pompeu Fabra
Roc Boronat 138, 08018 Barcelona Spain
sergi.jorda@upf.edu

1 Summary

People are continuously solving problems in their everyday activities. Some of these are "routine" problems that are easy to solve and have obvious and well-known criteria for identifying the solution by applying knowledge directly. Other problems are considered “complex” or “intractable” problems which people are unable to easily come up with a solution even if they are considered to have an adequate level of intelligence. In this case, divergent thinking and eventually creativity can make a difference in devising new solutions.

Creativity is therefore important for learning and future personal development, especially in the case of children and teenagers, and as a consequence it is also relevant for the whole society. How it can be fostered as well as evaluated in Information and Communications Technologies (ICT) settings seems to be a key issue for research. Despite advanced technology is being used to provide ICT systems according to creativity theories they rarely assess creativity itself, what brings up doubts on whether technologies actually provide some benefit in the expected direction. Moreover most of the computer-mediated approaches used to address creativity have been designed to support single-user interactions and so fail to consider other important dimensions of creativity, such as collaboration, reflection and divergent thinking in group face-to-face scenarios.

In this sense, the research presented in this paper is motivated by the expectation that tabletop technology and the evaluation of creativity will lead us to a better understanding of the creative process itself and will allow us to generate better creativity support on tabletop-based computer systems in the future. Hence, with the aim of exploring if interactive surface technology as base technology for collaborative creative tasks with teenagers is promising in terms of both collaboration and creativity

A. Ruíz, L. Iribarne (Eds.): JISBD’2012, pp. 201-202, ISBN:978-84-15487-28-9.
Jornadas SISTEDES’2012, Almería 17-19 sept. 2012, Universidad de Almería.

issues, this paper contributes by conducting an empirical study that measures creativity on two tabletop settings as an approach to evaluate how the environment can influence creativity.

In the experiment, teenager subjects were faced to a problem consisting of creating Rube-Goldberg machines (RGMs), which are mechanical systems mainly composed of building blocks connected to actionable devices, normally providing a complex solution to a simple problem. In particular, they were requested to design creative RGMs to solve a problem consisting of making a box fall from a shelf located in the center of the tabletop.

The study compared a tool implemented in interactive tabletop technology versus a completely physical and traditional tangible setting (i.e. with no computer mediation). The digital platform consisted of an interactive tabletop operated by means of multi-touch input and pucks. The physical-only platform relies on a conglomerate tabletop and a toolbox with wooden blocks and connecting elements. The choice of using a pure tangible platform instead of one based on a desktop application was made on the assumption that, firstly, this non-technological platform is similar to some construction kits widely used during childhood, and secondly, it is better to have two similar platforms in terms of co-located user involvement and participation possibilities.

The involved creativity assessment model considers a representative core set of features used in the psychology field, but with the idea that these can be to some extent impacted by the environment in which subjects interact. This model contains novelty, fluency and flexibility of thinking, elaboration, and motivation. The most important trait is undoubtedly novelty, which is defined as the characteristic conferring something unusual, unique or surprising.

In terms of creativity traits, interactive surfaces seem promising, as groups working on the digital platform showed significantly better performance in fluency of thinking, and motivation. Novelty was found to be near to significance. Other issues related to collaboration and interaction were also analyzed, including co-operation, retrial fine adjustment and dominance, which showed that the properties of an interactive surface tabletop are better suited to facilitating the sharing of objects and participation in conditions of co-operation by co-located participants. This research is therefore relevant for the interaction design of future tabletop-based information systems with collaborative requirements which primarily consider constructive tasks.

Acknowledgments

This work was funded by the Spanish Ministry of Education under project TSI2010-20488. Our thanks to the Alaquas city council, the clubhouse's managers, and also to Polimedia for the support in computer hardware. A. Catalá is supported by a FPU fellowship with reference AP2006-00181.

This communication is a summary of the following published article:
Alejandro Catala, Javier Jaen, Betsy van Dijk, and Sergi Jordà. 2012. Exploring tabletops as an effective tool to foster creativity traits. In Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction (TEI '12), Stephen N. Spencer (Ed.). ACM, New York, NY, USA, 143-150.
DOI=10.1145/2148131.2148163 http://doi.acm.org/10.1145/2148131.2148163

Desarrollo Dirigido por Modelos en Ingeniería Web con Webratio y RUX-Tool [#]

Rober Morales-Chaparro, Fernando Sánchez-Figueroa, Juan Carlos Preciado, Marino Linaje Trigueros, Roberto Rodríguez, José María Conejero,Pedro Clemente*
Quercus Software Engineering Group. Universidad de Extremadura. Cáceres, 10003
{*ponente: jcpreciado@unex.es}

Fernando Sánchez-Herrera, Alberto Díaz, Miguel Ángel Preciado, Javier García, Francisco Hermoso, Aquila Luque
Departamento de I+D+I. Homeria Open Solutions. Edificio Tajo – Campus Universitario. Cáceres, 10001
{fsherrera@homeria.com}

Resumen. A lo largo de los últimos años, se han planteado diferentes propuestas para la creación de aplicaciones Web en base a modelos conceptuales. Estos modelos tienen como objetivo principal el desarrollo de Webs basadas en grandes cantidades de datos (conocidas como Data Intensive Web Applications). Este tipo de modelos tiene en cuenta los diferentes elementos de información, organizados generalmente en páginas enlazadas que permiten estructurar los elementos y la navegación de la aplicación. Las modernas técnicas de generación automática de código sobre la base de diseño dirigido por modelos simplifica las fases más costosas del proceso de desarrollo de este tipo de aplicaciones (codificación, revisión y mantenimiento), reduciendo el uso de los recursos técnicos y humanos que se emplean y mejorando en algunos casos la calidad del producto final. En este sentido, la casi totalidad de estos modelos disponen de diferentes representaciones que permiten expresar los conceptos implicados en el diseño y desarrollo de una aplicación Web utilizando la noción de capas encapsuladas que dividen los objetivos en diferentes niveles según la responsabilidad requerida. Mediante esta división, cada una de las capas puede ser especificada de manera independientemente, definiendo en cada una modelos como pueden ser los de datos, navegación, etc.

Palabras clave: Ingeniería Web, Rich Internet Applications.

[#]*Este trabajo ha sido desarrollado en el contexto del proyecto MIGRARIA - TIN2011-27340, del Ministerio de Ciencia e Innovación y por el Gobierno de Extremadura, y del proyecto BPM4People, financiado por el REA de la Comisión Europea del Séptimo Programa Marco FP7 en la modalidad SME Capacities.*

1. Objetivo

Este tutorial está orientado a mostrar el funcionamiento y capacidades de las herramientas de desarrollo dirigido por modelos en Ingeniería Web como WebRatio (http://www.webratio.es) y RUX-Tool (http://www.homeria.com). Estas herramientas reducen el tiempo de desarrollo y los costes asociados, ampliando la capacidad de los desarrolladores y mejorando la eficiencia de soluciones Web.

WebRatio es una herramienta desarrollada y distribuida desde el año 2001 por la empresa WebModels S.L.R., spin-off del Politecnico di Milano (Italia). Proporciona un entorno de desarrollo vanguardista

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 203-206, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

integrado con Eclipse que permite el desarrollo de aplicaciones Web en base a modelos, permitiendo la generación automática de código para la plataforma J2EE.

RUX-Tool, herramienta desarrollada por Homeria, una spin-off de la Universidad de Extremadura, desde el año 2007, permite la generación automática de código para interfaces de usuario sobre diferentes tecnologías basadas en Rich Internet Applications tales como AJAX. Es una herramienta pensada para diseñadores Web que permite reducir considerablemente el tiempo dedicado a las labores de diseño y mantenimiento.

2. Motivación y Audiencia esperada

La relevancia de este tipo de tutorial se extrae de la robusta y novedosa generación automática de aplicaciones mientras que proporciona valores óptimos de calidad, escalabilidad y *mantenibilidad*. Esta generación automática se basa en modelos conceptuales, métodos validados y motores de transformación y generación de código innovadores que permiten descubrir un rango de beneficio amplio dentro del escaso margen que define la actual relación entre clientes demandantes de aplicaciones web y las empresas desarrolladoras.

Este tutorial está orientado a investigadores e ingenieros de desarrollo de software vía Web, software y comunidad de JISBD en general.

3. Esquema de Contenidos

El tutorial se impartirá en modo resumen para ofrecer una panorámica completa en 90 minutos, donde los asistentes observarán los contenidos y ejemplos en la pantalla/proyector del profesor del mismo. Al final del tutorial se entregarán licencias de prueba de las tecnologías a los asistentes interesados. El tutorial se fundamenta en los siguientes puntos principales:

- Conceptos implicados en Ingeniería Web y Rich Internet Applications
- Despliegue de la tecnología
- Diseño de modelos de datos
- Diseño de modelos de lógica de negocio
- Diseño de presentación RIA con RUX-Tool

4. Presentación del tutorial

Los participantes recibirán varios días antes del mismo un manual en formato electrónico para agilizar el desarrollo de aplicaciones con las herramientas a presentar. En dicho manual se presenta y describe la tecnología, los conceptos básicos implicados así como el ejemplo que se seguirá en el tutorial.

La baja curva de aprendizaje que estas herramientas ofrecen brinda la posibilidad de transmitir los conceptos implicados de una forma directa en el tiempo dedicado al tutorial.

Biografía resumida

- *Roberto Morales Chaparro*. Rober Morales-Chaparro es Ingeniero en Informática desde 2007 por la Universidad de Extremadura. Su principal línea de investigación incluye el Desarrollo de Software Dirigido por Modelos para Aplicaciones en la Web. Actualmente se encuentra realizando su doctorado acerca de técnicas de Ingeniería del Software aplicadas a Visualización de Datos y Business Intelligence.
- *Fernando Sánchez Figueroa*. Fernando Sánchez-Figueroa es profesor Titular del Departamento de Ingeniería de Sistemas Informáticos y Telemáticos de la Universidad de Extremadura. Es co-fundador de la spin-off Homeria Open Solutions, S.L. empresa desarrolladora de RUX-Tool y que comercializa para España Webratio. Actualmente participa en dos proyectos del VII Programa Marco de la UE donde se aplica esta tecnología, CUBRIK y BPM4PEOPLE.
- *José María Conejero Manzano*. Es profesor Ayudante Doctor del área de Lenguajes y Sistemas Informáticos de la Universidad de Extremadura. Sus principales líneas de investigación son la Ingeniería Web, el Desarrollo de Software Dirigido por Modelos, el Desarrollo de Software Orientado a Aspectos y la Inteligencia Ambiental.
- *Juan Carlos Preciado Rodríguez*. Doctor en Ingeniería Informática por la Universidad de Extremadura. Desde hace 10 años, es profesor de universidad, impartiendo las asignaturas de Ingeniería del Software y Web. Trabaja en la temática de Ingeniería Web para el desarrollo de métodos de soporte a la generación automática de aplicaciones ricas de Internet en dirigida por modelos. Trabaja en Business Intelligence, Data Visualization y Web Engineering, además en varios proyectos internacionales relacionados con su temática de investigación.
- *Marino Linaje Trigueros.* Doctor en Ingeniería Informática por la Universidad de Extremadura. Desde hace 8 años, ejerce como profesor de la UEx. Tiene experiencia como investigador, lo que le ha permitido conocer perfectamente el sector de las TIC y mantener contactos con universidades y entidades europeas. En cuanto a su experiencia empresarial, tiene conocimientos en el ámbito del desarrollo Web, en el contacto con clientes y en creación, análisis y desarrollo de soluciones.
- *Pedro José Clemente Martín.* Ingeniero en Informática y Doctor en Informática por la Universidad de Extremadura. Profesor del Área de Lenguajes y Sistemas Informáticos de la Universidad de Extremadura desde 2000. Como Investigador su interés se centra en las Arquitecturas Orientadas a Servicios, el desarrollo de Software basado en Componentes y el Desarrollo de Software Dirigido por Modelos.
- *Francisco Hermoso Baños*. Ingeniero en Informática por la Universidad de Extremadura. Consultor experto en el desarrollo Front-end de aplicaciones web. Analista-Programador en el desarrollo de RUX-Tool. Desde 2009 trabajando en Homeria con WebRatio/RUX-tool y las tecnologías MDD y Single Page.
- *Alberto Díaz Díaz*. Ingeniero Técnico en Informática de Sistemas (2009) e Ingeniero en Informática (2011) por la Universidad de Extremadura. Desde 2009 analista, formador y consultor WebRatio en Homeria Open Solutions S.L.
- *Javier García Morón*. Ingeniero Técnico Informático de Sistemas por la Universidad de Extremadura desde 2008. Máster en Formación del Profesorado. Consultor - desarrollador de Rich Internet Applications en Homeria desde 2009.
- *Miguel Ángel Preciado*. Ingeniero Técnico en Informática por la Universidad de Extremadura. Máster en Software Libre. Ha participado como investigador de RUX-Method en la Universidad de Extremadura y el desarrollo de la herramienta RUX-Tool. Actualmente es el director de operaciones del proyecto internacional CUbRIK en Homeria Open Solutions.

- *Aquila Luque Jáñez.* Ingeniero en Informática (2008) e Ingeniero Técnico en Informática de Sistemas (2005) por la Universidad de Extremadura. Master en Software Libre (2009) cursado en la Universidad de Extremadura.
- *Fernando Sánchez-Herrera.* Ingeniero técnico en informática de sistemas por la Universidad de Extremadura (2002-2005), designado mejor expediente del año en su titulación. Master en software libre, especialidad desarrollo por la Universidad de Extremadura (2009). Ha trabajado en proyectos de investigación de la Universidad de Extremadura, centrándose en el campo de los discapacitados visuales. Socio fundador de Homeria Open Solutions, empresa de la que actualmente es director general.
- *Roberto Rodríguez-Echeverría.* Profesor Colaborador del Departamento de Ingeniería de Sistemas Informáticos y Telemáticos de la Universidad de Extremadura. Más de 10 años de experiencia docente. Ha ocupado diferentes cargos de gestión TI en la institución durante los últimos 8 años, siendo actualmente el Subdirector Técnico del Campus Virtual. Sus principales líneas de investigación son la Ingeniería Web (RIA), el Desarrollo Dirigido por Modelos, la Ingeniería Inversa y la Reingeniería de Sistemas Heredados.

Sesión Temática 3

Apoyo a la decisión en Ingeniería del Software, Metodologías, Experimentación

Coordinadores: *Dra. Mercedes Ruiz y Dr. Agustín Yagüe*

Sesión Temática 3: Apoyo decisión Ing. Software, Metodologías, Experimentación
Coordinadores: Dra. Mercedes Ruiz y Dr. Agustín Yagüe

Daniel Crespo and Mercedes Ruiz. *SIM4CMM: Decision Making Support in CMMI Based Project Management*. (Herramienta)

Tomas Martinez-Ruiz, Felix Garcia and Mario Piattini. *SPRINTT: Un Entorno para la Institucionalización de Procesos Software*. (Regular)

Andrea Delgado, et al. *Un experimento para validar transformaciones QVT para la generación de modelos de servicios en SoaML desde modelos de procesos de negocio en BPMN2*. (Regular)

Carlos López, M. Esperanza Manso and Yania Crespo. *Evaluación de la eficiencia en métodos de identificación del defecto de diseño God Class*. (Regular)

Raúl Marticorena and Yania Crespo. *Alf como lenguaje de especificación de refactorizaciones*. (Regular)

Ana M. Moreno, Agustín Yagüe and Diego Yucra. *Usability mechanisms extension to ScrumTime*. (Herramienta)

Ana M. Moreno, Agustin Yague and Diego Yucra. *Tailoring user stories to deal with usability*. (Regular)

Jose Antonio Cruz-Lemus, et al. *Réplica de un experimento que estudia las relaciones extroversión-calidad y extroversión-satisfacción en equipos de desarrollo de software*. (Regular)

Isabel María Del Águila, José Del Sagrado and Francisco Javier Orellana. *Metaheurísticas como soporte a la selección de requisitos del software*. (Regular)

Jose Antonio Cruz-Lemus, et al. *Assessing the Influence of Stereotypes on the Comprehension of UML Sequence Diagrams: A Family of Experiments*. (Relevante)

A. Ruíz, L. Iribarne (Eds.): Actas de las "*XVII Jornadas de Ingeniería del Software y Bases de Datos (JISBD'2012)*", Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

SIM4CMM: Decision Making Support in CMMI Based Project Management

Daniel Crespo and Mercedes Ruiz

University of Cadiz,
C/ Chile, 1, 11003 - Cadiz, Spain
dani.crespobernal@alum.uca.es,
mercedes.ruz@uca.es

Abstract. Estimates of task duration or the amount of resources needed in software projects are often very inaccurate. To avoid this problem, project management must be effective and dynamic, that is, being proactive rather than reactive. Among the tasks needed in this approach, reassigning resources, hiring new personnel or adapting estimates to new situations can be found. We propose a tool to help decision making in project management offering a real time simulation of the project team behavior, the interaction with the project tasks and the project metrics.

Keywords: Multiparadigm Simulation Modeling; Project Planning; CMMI; Hybrid models;

1 Introduction

Now, more than ever, organizations want to develop products and services in an optimal, fast and inexpensive way. At the same time, in the 21st century high technology sector, almost every organization has found itself developing products and services of increasing complexity. The problems these organizations face nowadays need solutions that require the involvement of the entire company and an integration approach. Effective management of organizations assets is essential for business success.

CMMI (Capability Maturity Model Integration) consists of best practices for the development and maintenance activities of products and services [1]. CMMI can be used as: a) a best practice collection in process improvement activities, b) a framework to prioritize and organize activities, c) a support to coordinate multidisciplinary activities to properly build a product and/or d) a way to align improvement process goals with organization business goals. In order to support the design and execution of the Specific Practices (SP) described in CMMI Project Planning (PP) and Project Monitoring and Control (PMC) process areas we propose a simulation tool which may be used by the management of an organization as a decision making assistant in critical situations during the execution of a software project. SIM4CMM has been validated using the data from the International Software Benchmarking Standard Group (ISBSG) v. 10 project repository [2]. Before using it in a software organization it is always necessary to calibrate SIM4CMM according to the organization's historical data.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 209-212, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

2 Goals

SIM4CMM is a tool that simulates the life cycle of a software project enabling users to experiment with different scenarios customizing the parameters of the project as well as the decisions made in a particular situation. The main goal of SIM4CMM is to help in decision making in project management before making decisions in the real world. To make this possible SIM4CMM obtains the estimated metrics for the project and shows graphically the whole process of the project step by step. SIM4CMM supports the following actions: a) Optimize the number of resources used in a project maximizing the profit margin of the organization, b) Simulate different resources allocation strategies, c) Simulate the project with different life cycles and d) Allow the simulation of risk management politics and the implementation of corrective actions.

The potential users of SIM4CMM are Project Managers in software companies and organizations. However, this tool could be used in higher strategy levels since future versions would include new CMMI areas in order to achieve higher CMMI levels. SIM4CMM could also be used in academic field as a training tool for future Project Managers or in higher university courses.

3 Design & Development

The main contribution of SIM4CMM consists in using a multiparadigm simulation model integrating discrete event and agent-based simulation approaches to model concrete parts of the development process in software projects. Both project coding and project testing processes have been simulated by agent-based models, thus obtaining a more realistic vision and a much more accurate estimation when modeling the project team behavior from the point of view of the people involved. This model supports the specific practices of the PP and PMC CMMI process areas emphasizing schedule fixing, effort estimation, adequate life cycle election, project total cost estimation and resources number needed to develop the project. The model implementation and the simulation runs have been performed using Anylogic^{TM}simulation software. The model logic is written in Java and the databases have been designed and implemented using MySQL.

As stated before, the project teams have been modeled using the agent-based paradigm. Every member of the team is an agent and has a set of descriptive parameters. These agents interact with tasks contributing to their completion taking into account agent parameters. For example, one programmer can be a better team worker than another, therefore his productivity may be higher. The parameters used to model an agent follows: a) Team Work Influence, b) Adaptation, c) Proactivity, d) Commitment, e) Programming Language Skills and f) Development Methodology Proficiency. SIM4CMM allows to study the effects of different types of resource allocation strategies within the coding and testing processes. Two strategies are analyzed: a) While there are idle resources, they are assigned to new tasks of the pending tasks set, and b) While there are idle resources, determine if a task needs an extra support in order to finish on time. If so, assign the resource to the running task.

In order to support the Specific Practices described in CMMI PMC process area, an inspection team can be simulated in order to detect early errors in code and design tasks. The Project Manager can evaluate the pros and cons of using the inspection process in the project from the information obtained after simulation. To do this, the user must provide the values for some input parameters such as: Inspection Effort (Effort made by Inspection Team), Inspection Efficiency (Ratio of efficiency of inspectors), Rework Effort (Rate associated to solve errors, measured in hours), Error Generation Density, Final Rework Penalty (Penalty associated to redo tasks in final project phases), etc.

4 Usage Example

The user can easily simulate a project using the Graphical User Interface (GUI) partially depicted in Figure 1. In the first place, the database of the simulation tool must contain all the information related to the tasks being developed during the project as well as the information about agents and specific project data (size, budget, schedule, etc). Since we are done with this step, we can configure the simulation parameters for the particular situation or project. The main control parameters are: a) Life Cycle, b) Resources Allocation Stategy and c) Inspection Process Enablement. Once we start the simulation, the GUI will show the following information: a) Clock, showing simulation date and time, b) Real-Time project metrics, c) Development Team Status, d) Testing Team Status, e) Effort charts, f) Project Sequence and g) Project evolution over time. The user can vary the time step in order to run the simulation faster or slower depending on the current project phase or user interest. When the simulation ends the user obtains the following metrics referring to the simulated project: a) Estimated cost of the project, b) Estimated completion time, c) Estimated profit margin (the result of substracting direct costs of the project to the economic benefits obtained) and d) Estimated number of defects obtained. Also the effort distribution of the project is presented graphically and numerically in order to study the effect of the strategies selected and decisions made during simulation.

5 Conclusions and Future Work

In this paper we present a simulation based tool to help project management in CMMI organizations as well as achieving process improvement. Specifically, a multiparadigm simulation model in the realm of CMMI PP and PMC process areas has been developed in order to support the specific practices of the areas aforementioned. This has been done by simulating the development process of a project from planning to deployment starting from project information known by the organization which has been modeled.

As a future work, we aim to simulate different alternatives in particular process area construction in order to determine which one fits better to the organization, study the process behavior over time and offer simulation-support for CMMI Level 4 process areas building and quantitative project management.

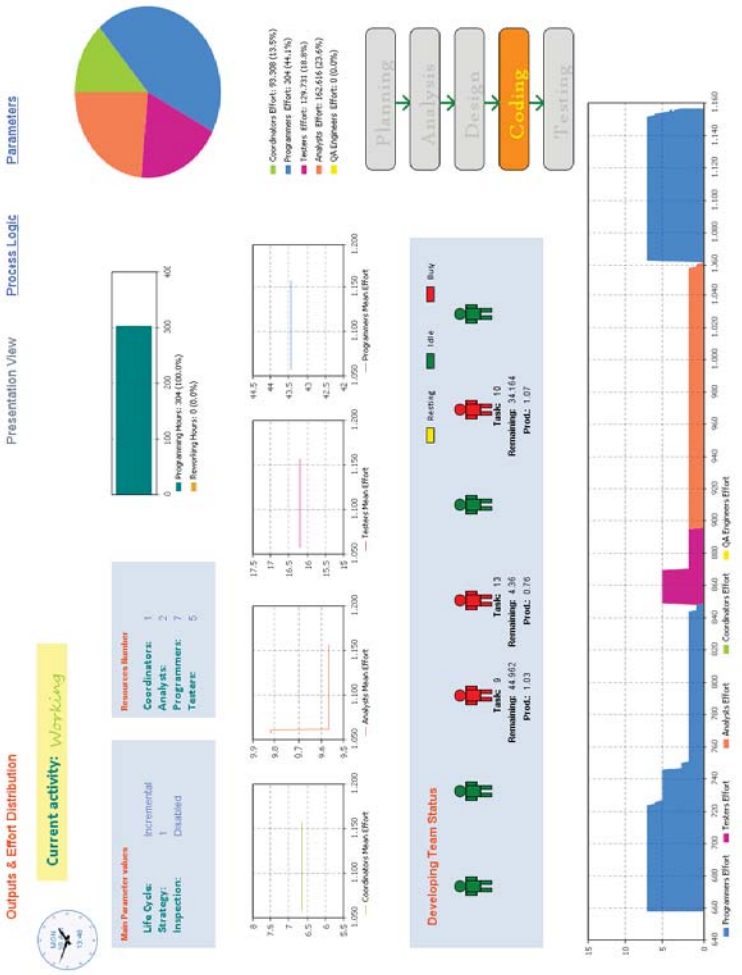


Fig. 1. Project Management Decision Making Tool GUI

Acknowledgments

This research is partially funded by the Ministry of Education and Science of Spain and European FEDER funds under project TIN2010-20057-C03-03.

References

1. CMMI Product Team: CMMI for development, version 1.2. Tech. Rep. CMU/SEI-2006-TR-008, Software Engineering Institute (aug 2006), `http://www.sei.cmu.edu/reports/06tr008.pdf`
2. The International Software Benchmarking Standards Group: ISBSG repositories (2007)

SPRINTT: Un Entorno para la Institucionalización de Procesos Software

Tomás Martínez-Ruiz, Félix García, Mario Piattini

Instituto de Tecnologías y Sistemas de Información.
Universidad de Castilla-La Mancha.
Ciudad Real, España
{tomas.martinez, felix.garcía, mario.piattini}@uclm.es

Abstract. La adaptación de procesos es una tarea crucial. Sin embargo, no es sencillo hacer cambios de forma ad-hoc dentro de un proceso y esperar que sea correcto y consistente. Cualquier organización se enfrenta continuamente a este reto cuando lleva a cabo sus proyectos de acuerdo a sus modelos de procesos teniendo en cuenta las características de cada proyecto. Como resultado, se obtienen versiones del modelo de procesos que cada vez es menos manejable ante los múltiples cambios realizados de forma ad-hoc. La solución pasa por dotar a los procesos software de mecanismos adecuados para la adaptación sistemática y además aprovechar el conocimiento obtenido en cada adaptación para mejorar el propio modelo de procesos. Con todo ello en este artículo se presenta el ciclo SPRINTT para la institucionalización de procesos software que promueve la adaptación y estandarización de variantes y el paradigma de Procesos Ricos en Variantes (VRP) en el que se basa. El paradigma integra la variabilidad dentro de los procesos, para adaptarlos según cada proyecto, de manera sencilla y consistente. La propuesta se ha aplicado en un caso de estudio para la definición de procesos adaptables de Desarrollo Global de Software. Finalmente se propone extender este enfoque a nivel de contexto para vincular cambios en la organización y variaciones dentro de un proceso rico en variantes.

Keywords: Adaptación de procesos, Variabilidad, Procesos ricos en variantes, Rationale Management

1 Introducción

Tal como establece Fuggetta [1], la calidad del software depende de la capacidad de los procesos de desarrollo y mantenimiento, desde entonces no han sido pocas las iniciativas enfocadas en proporcionar modelos de procesos "capaces" para garantizar la calidad del software durante su desarrollo. Estos esfuerzos se han materializado en propuestas como CMMI, ISO 12207, que aunque incluyen las mejores prácticas, debido a la genericidad propia de los estándares, no reflejan la realidad específica de la organización en la que se quiere implantar.

De acuerdo con la *Teoría de la Evolución*, en la naturaleza sólo sobreviven aquellos seres vivos que incluyen mutaciones que les hace adaptarse mejor al medio. Siguien-

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 213-226, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

do ésta filosofía, en el área de procesos software tenemos que seleccionar aquellos procesos que incluyen, en este caso variaciones, que les hacen adaptarse mejor al contexto definido por la organización y proyectos dados, y por tanto identificar estas variaciones satisfactorias. En la literatura se pueden encontrar diversos trabajos centrados en la adaptabilidad o variabilidad de los procesos software, entre estos destacan las propuestas de Simidchieva et al [2] o Denger et al.[3].

Sin embargo, estas iniciativas sólo proporcionan soporte a la adaptación de procesos, en ningún caso se utilizan para institucionalizar el proceso software dentro de la organización, que según Chrissis et al. [4] consiste en hacer que el proceso se integre, utilice y constituya una guía real del trabajo. Con el fin de dar respuesta a esta necesidad, en este artículo se presenta el Entorno para la Institucionalización de Procesos Software (SPRINTT). Con él se pretende facilitar la adaptación de procesos software siguiendo el Paradigma de Procesos Ricos en Variantes (procesos con diversas variantes), así como la estandarización de las variantes que proporcionan una correcta adaptación del proceso para cada contexto. De esta manera se consigue que cada vez que se adapta un proceso software, además de conseguir el propio proceso adaptado, el proceso encaja mejor dentro de la organización. En concreto, este artículo presenta una visión general del entorno y del paradigma de Procesos Ricos en Variantes, junto con su lenguaje de soporte vSPEM, así como los resultados de su aplicación en un caso de estudio.

Además de esta introducción, la sección 2 se centra en la descripción del entorno y del paradigma, mientras que la sección 3 describe un caso de estudio de la aplicación de vSPEM sobre un proceso real. En la sección 4 se presentan las propuestas más relevantes. Finalmente las conclusiones y trabajo futuro se presentan en las secciones 5 y 6 respectivamente.

2 Institucionalización de Procesos Software

Los modelos de referencia de procesos incluyen niveles de capacidad para denotar su grado de consecución y guiar en su desempeño, partiendo desde la mera aplicación del proceso hasta que está perfectamente integrado. Es común en las escalas de capacidad que uno de sus niveles se defina como que el **proceso además de ejecutarse dentro de la organización está lo suficientemente maduro como para adaptarse a partir de un conjunto de procesos estándares de la propia organización**. Aunque el concepto aparece en todos los modelos de procesos, dentro del contexto de CMMI, se le conoce como proceso *Institucionalizado* [4], y a la obtención de estos procesos, *institucionalización de procesos*

Por su definición, un proceso institucionalizado es aquel que encaja perfectamente dentro de la organización y puede ejecutarse sin resultar una carga. Pero institucionalizar un proceso no es trivial, por una parte es necesario adaptar cada proceso a las circunstancias exactas de ejecución, pero sin perder de vista su propósito y objetivos originales. Por otra parte las adaptaciones deben ser registradas junto con el contexto exacto en el que se aplican para obtener conocimiento que pueda reutilizarse en posteriores adaptaciones.

Para dar soporte adecuado a la institucionalización de procesos se ha diseñado el entorno SPRINTT. Este constituye un marco integrado que soporta y combina de forma adecuada ambas técnicas, de adaptación y estandarización de procesos software. El entorno se articula en torno al ciclo de institucionalización de procesos software, e integra el Paradigma de Procesos Ricos en Variantes para adaptar los procesos.

2.1 Ciclo de Institucionalización de Procesos Software.

Los procesos software son entidades "vivas" por tanto ningún enfoque puede gestionarlos de manera eficiente si no es cíclico, tales como por ejemplo las propuestas de Arbaoui et al. [5] o IDEAL [6]. Por ello, el ciclo de institucionalización consta de cuatro fases, tal y como muestra la Figura 1:

1. **Adaptación**: durante esta fase, el proceso es adaptado mediante variaciones, utilizando los activos disponibles (conocimiento previo) dentro de la organización, a la vez que se guarda un registro detallado de cómo se adapta el proceso y de las alternativas y decisiones que llevan a cada una de las acciones de adaptación. De esta forma existe una traza completa entre las necesidades del contexto y las variaciones que las satisfacen.
2. **Ejecución**: Durante esta fase el proceso se ejecuta en el proceso dentro de la organización y proyecto para los que se adaptó, y quedan registradas las variaciones o desajustes de la adaptación de la fase anterior.

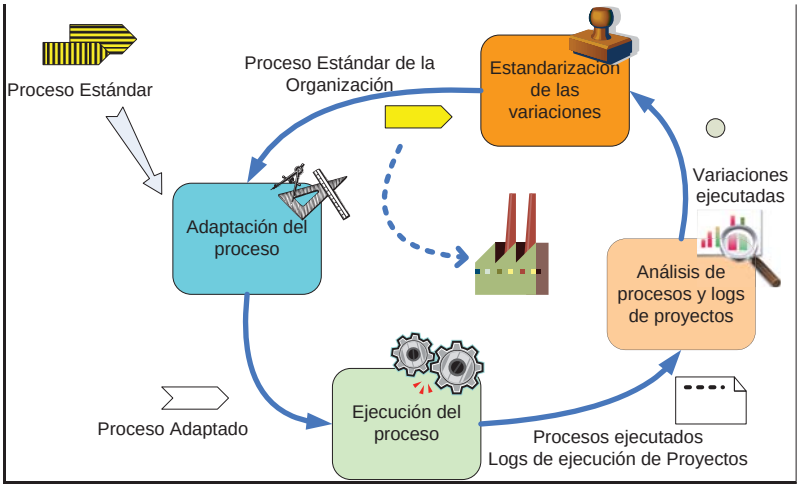


Figura 1. Ciclo de Institucionalización de Procesos Software

3. **Análisis**. En esta fase se analiza y comparan *postmortem* las variaciones realizadas con el comportamiento esperado de éstas en el proceso. De esta forma se pueden filtrar las variaciones bien realizadas y que son susceptibles de volver a ejecutarse en posteriores adaptaciones de procesos.

4. **Estandarización**. Durante esta fase se vinculan las variaciones bien realizadas extraídas en la fase de análisis con el contexto para el que se realizaron. De esta forma se estandariza su uso y facilita su reutilización en proyectos similares.

En resumen, el ciclo de institucionalización es capaz en cada iteración de acercar un poco más el proceso hacia la realidad que se vive dentro la organización, es decir, institucionalizarlo. La Figura 2 muestra la infraestructura necesaria para soportar el ciclo. Por una parte, los repositorios que almacenan los activos generados en cada una de las fases, y por otra se hace patente la necesidad de dos nuevos elementos. El primero de ellos es un conjunto de mecanismos que den soporte a la adaptabilidad de los procesos software, de una forma bien definida y acotada (véase Sección 3.2). El segundo es la gestión de Rationale, puesto que la adaptación de procesos, cualesquiera que sean los mecanismos con los que se lleva a cabo, necesita estar bien fundamentada, razonada y ser trazable, máxime cuando se pretende extraer conocimiento de ella.

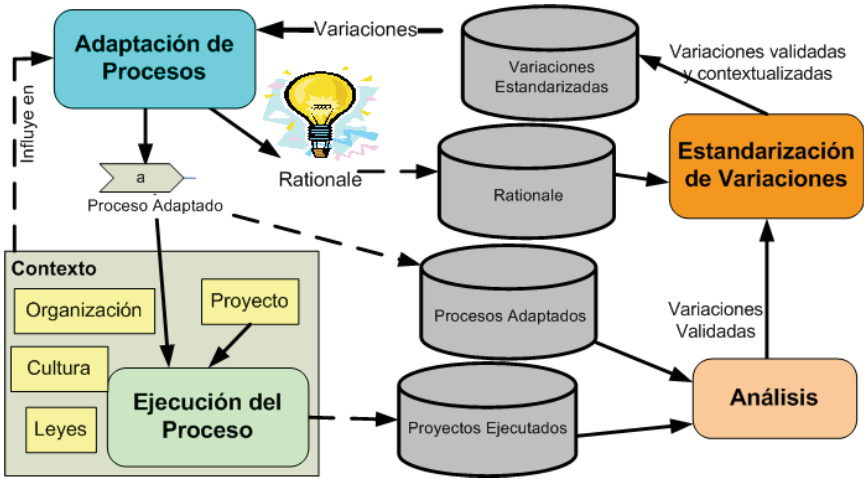


Figura 2. Detalle de la infraestructura de Institucionalización de Procesos

2.2 Paradigma de Procesos Ricos en Variantes

El paradigma de Procesos Ricos en Variantes (VRP) nació con el objetivo de dar soporte a la variabilidad de procesos software por una parte tal y como la revisión sistemática demostró que la industria necesitaba [7], y por otra para utilizarse dentro del entorno de institucionalización. Los resultados de la revisión evidenciaron que en la actualidad no existe una única notación, y que todas ellas se utilizan y por ende tienen la necesidad de soportar la adaptación de procesos.

Ante esta demanda se diseñó el paradigma como el compendio mínimo de elementos, constructores y reglas que una notación necesita incorporar para dar soporte a la variabilidad de procesos software. Estos constructores se han dividido a su vez en dos paquetes, *variations* y *vRichProcess*. El primero de estos paquetes contiene sólo los

elementos que soportan la variabilidad de procesos y es perfectamente útil para aquella organización que implemente la primeras dos fases del ciclo de institucionalización. Dentro del segundo paquete están los elementos que gestionan el conocimiento sobre las variaciones que se utiliza en la tercera y cuarta fases del ciclo. Con esto, el Paradigma intenta adaptarse a las necesidades que cada organización.

El paradigma ha sido diseñado en base a tres grandes pilares:

1. **Variaciones puntuales** [8, 9]. Son las que afectan únicamente a un elemento del proceso cada vez (aparte de que por consistencia tengan que realizarse otras variaciones). Por ejemplo en el caso de eliminar un producto de trabajo optativo, o de cambiar el perfil de un rol. Se han soportado dentro del paradigma a través del uso de variantes y puntos de variación importados del enfoque de Líneas de Producto Software [10].
2. **Variaciones transversales** [11]. Estas variaciones satisfacen la necesidad de cambios a gran escala dentro del proceso, teniendo en cuenta que los procesos software son grandes estructuras compuestas por elementos atómicos. Por ejemplo, la adición de tareas de validación antes de la entrega (final) de cada entregable, o la sustitución de determinadas tareas por otras similares, de acuerdo al uso de uno u otro estándares de seguridad. Las variaciones transversales son una agrupación ordenada de variaciones puntuales, que como en el caso de los procedimientos en SQL se ejecutan todos o ninguno. Se han descrito utilizando los activos que propone la Ingeniería de Software Orientada a Aspectos [12]. La Tabla 1 muestra las equivalencias entre los elementos de AOSE y del paradigma..
3. **Gestión de Rationale** [13]. Esta parte del paradigma no da soporte directamente a la adaptación de procesos, sino que por el contrario lo complementa para guiar en la toma de decisiones para hacer un mejor uso de las variaciones puntuales y transversales, y por otra parte facilita la extracción y reutilización de conocimiento en la ejecución de nuevas variaciones. Para ello se adapta la propuesta de [14] para dar soporte a la toma de decisión en a la adaptación de procesos software.

Tabla 1. Equiparación de los elementos de AOSE utilizados en el paradigma

Concepto AOSE	**Elemento de línea de procesos**	**Tipo de variación**
Join point	Variation point	Puntual
Advice	Variant	
Point cut	Relaciones entre los puntos de variación y los advices	
Crosscutting Concern	Crosscutting Variation	Transversal
Aspect	Aspect	
Point cut	Filter	

El paradigma incluye los constructores para la gestión de variabilidad en procesos ricos en variantes. Sin embargo, no define estos constructores, pues al ser el paradig-

ma genérico, los constructores concretos deben especificarse en base a los constructores específicos de proceso de la notación a la que se quiera dotar con la capacidad de variación. En este caso, el paradigma se ha utilizado para dotar de variabilidad al estándar SPEM obteniendo como resultado una extensión denominada vSPEM.
vSPEM

2.3 vSPEM

El lenguaje vSPEM nace como una implementación del paradigma de Procesos Ricos en Variantes sobre SPEM [15], y para paliar las deficiencias en variabilidad de procesos que éste presenta. Con todo ello, la estructura de *vSPEM* se compone de 7+2 paquetes. A los 7 originales aportados por SPEM, añade dos, *Variations* y *VRichProcess*. Estos paquetes implementan las funcionalidades descritas en el paradigma, y dotan a este con la capacidad de modelar variabilidad en procesos software y gestionar procesos ricos en variantes, respectivamente:

1. Las variaciones puntuales se soportan mediante la definición de puntos de variación y variantes de todos los constructores de proceso, que incluyen una notación gráfica para facilitar la lectura de los diagramas.
2. Las variaciones transversales definen una gramática propia que permite navegar por el proceso rico en variantes, y filtrar y ejecutar variaciones sobre los elementos requeridos.
3. Finalmente, la gestión de Rationale se materializa a través de constructores que vinculan las necesidades de variación con las variaciones realizadas (Figura 3), optimizando el uso de los mecanismos de variación.

3 Caso de Estudio: ORIGIN

La validación de la notación vSPEM y por tanto del Paradigma de Procesos Ricos en Variantes sobre un lenguaje particular, se ha llevado a cabo mediante un caso de estudio según las recomendaciones de [16]. El contexto del caso de estudio lo ha constituido el proyecto de I+D+I ORIGIN, que aborda los problemas típicos que aparecen en proyectos de desarrollo global de software, tales como la dificultad de comunicación, coordinación, control y la necesidad de una adecuada gestión de conocimiento. Uno de los principales retos de este proyecto es dotar a las organizaciones de mecanismos que permitan adaptar sus procesos en un contexto de desarrollo global en el hay que considerar las particularidades de cada nodo o localización (factores culturales, organizacionales, procesos locales, etc..).

3.1 Diseño, Sujetos y Unidades de Análisis

La pregunta que dirige el caso de estudio es:
¿Es el paradigma de Procesos Ricos en Variantes adecuado (útil y práctico) para modelar variaciones a través del lenguaje vSPEM en procesos software reales?

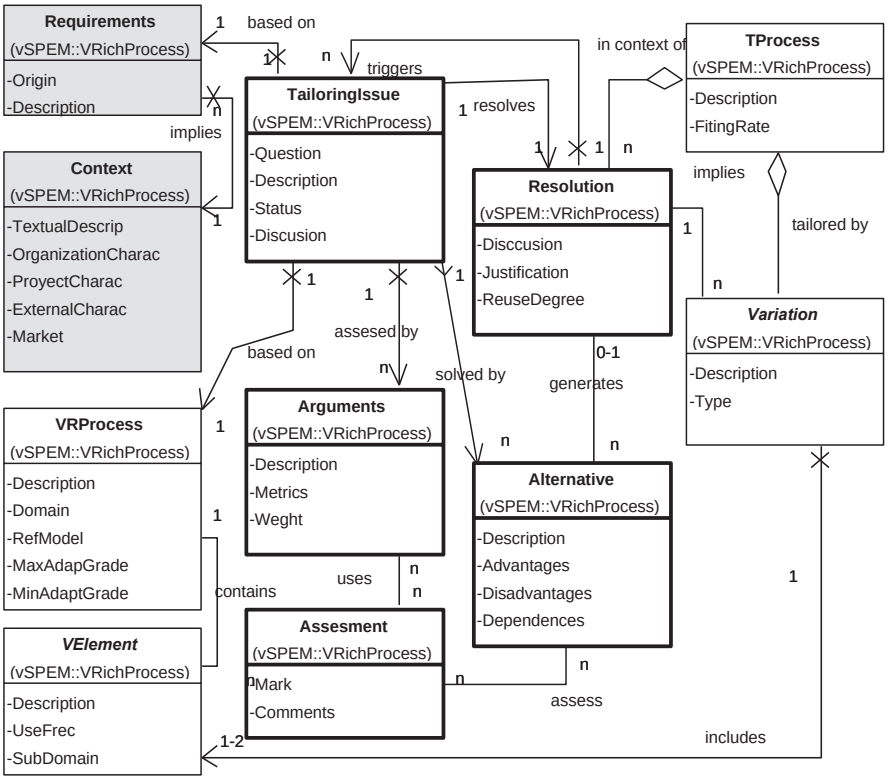


Figura 3. Aplicación del Rationale para la decisión de variaciones

La pregunta se enfoca en demostrar si la notación es adecuada para modelar las variaciones reales que aparecen en un modelo de procesos, y por tanto facilita su adaptación. A través del lenguaje se está comprobando si el paradigma es eficiente en el modelado de estas variaciones.

Respecto de la estructura del estudio, se usó un *diseño simple*, pues solo se usa un modelo de procesos (compuesto de varios procesos). El *objeto del estudio* es la notación vSPEM (y de forma indirecta el paradigma). Las *medidas* usadas en la investigación han sido el esfuerzo en modelar los elementos de variabilidad en el proceso.

En cuanto a los sujetos involucrados en el caso de estudio, éste fue ejecutado por el autor principal, con la supervisión de los otros autores que actuaron como expertos y los creadores de la metodología ORIGIN.

El análisis se enfocó sobre los procesos de la metodología, haciendo especial énfasis en las necesidades de variabilidad que éstos presentan de forma implícita, y en cómo los mecanismos descritos en vSPEM pueden modelar esa variabilidad.

3.2 Procedimiento y Recogida de Datos

Se aplicó un procedimiento orientado en encontrar y definir la variabilidad implícita de la metodología tal y como muestra la Tabla 2.

Tabla 2. Procedimiento para la ejecución del caso de estudio

1. Planificación de la variabilidad y Análisis de contexto
    a. Determinación del alcance del contexto
    b. Análisis sintáctico del modelo de procesos
    c. Análisis semántico
2. Diseño de las variaciones
    a. Definición de variaciones
    b. Definición de consistencia en las variaciones
3. Implementación de las variaciones
    a. Implementación de variaciones puntuales
    b. Implementación de variaciones transversales
4. Presentación del proceso rico en variantes resultante

3.2.1 Planificación de la Variabilidad

Una vez analizados los procesos de ORIGIN, se han determinado las siguientes necesidades de variabilidad.

1. Diferencia Horaria, o grado de solapamiento entre los horarios laborales de los grupos de trabajo.
2. Reuniones distribuidas, en cuanto a las diferentes formas de ejecutar las reuniones distribuidas.
3. Método de formación de los equipos distribuidos
4. Modo de gestión del *Equipo Distribuido*
5. Metodología de implementación y pruebas
6. Gestión de documentos compartidos
7. Necesidad de (la actividad) *retrospective de retrospectives*
8. Necesidad de la actividad *Reunión scrum of scrums*

3.2.2 Diseño e Implementación de las Variaciones

La Tabla 3 enumera los elementos de variabilidad utilizados en cada variación y define el tipo de variación (puntual o transversal). En la Figura 4 puede verse el diagrama correspondiente a la fase *Trabajo diario*. El diagrama muestra el proceso rico en variantes, junto con dos de las variantes diseñadas para soportar variabilidad, ambas están relacionadas con el aspecto *Documentación*. Este aspecto es una de las posibles implementaciones de la variabilidad en el *modo de comunicación*.

3.3 Análisis de Resultados

3.3.1 Esfuerzo

El esfuerzo utilizado en realizar el caso de estudio fue de una persona y semana. Sin embargo, gran parte del esfuerzo se concentró en el análisis del contexto y planificación de variabilidad, es decir, en buscar dentro de la metodología, aquellos ele-

mentos que pudieran introducir variabilidad, no en modelarla usando los mecanismos de variabilidad. Una vez descrita las necesidades de variabilidad, y utilizando la herramienta adecuada, el modelado no conllevó dificultad.

3.3.2 Adecuación

Tal y como se remarca en la Figura 4, los mecanismos de variabilidad dan soporte útil al modelado de las diferentes variaciones que se pueden ejecutar en el proceso. De igual forma, los mecanismos controlan la consistencia interna de los procesos ricos en variantes y los procesos adaptados.

Tabla 3. Elementos asociados a las variaciones puntuales y transversales

Variación	Tipo	Ptos Var.	Variantes	Aspectos
Diferencia horaria	Trans.	n ptos	n variantes	8 aspectos
Reuniones distribuidas	Trans.	n ptos	5n variantes	5 aspectos
Creación de equipo distribuido	Punt.	1 pto	3 variantes	
Gestión del equipo distribuido	Trans.	n ptos	4n variantes	4 aspectos
Met. de implementación y testing	Punt.	1 pto	2 variantes	
Gestión de docs compartidos	Trans.	n ptos	3n variantes	3 aspectos
Actividad de *Retrospective of retrospective*	Punt.	1 pto	1 variante	
Actividad de *scrum of scrums*	Punt.	1 pto	1 variante	

3.3.3 Soporte a la Adaptación

Una vez modelados los procesos ricos en variantes, se hizo una simulación de la adaptación de un proceso a partir de éstos. El resultado mostró que la adaptación de procesos se simplifica a elegir para cada opción de variabilidad aquella implementación que mejor se ajusta a las necesidades planteadas, bien sea utilizando los variaciones puntuales o transversales.

3.4 Análisis de Validez y Limitaciones

Para disminuir los riesgos que amenazan a la validez de constructo, se utilizaron plantillas específicas para cada uno de los constructores en los que se han definido claramente los elementos de variabilidad, sus propiedades y relaciones. Los riesgos de validez interna se minimizaron mediante la definición de un procedimiento detallado para la obtención y modelado de las variaciones, de igual forma todo el proceso ha sido supervisado por los expertos en el área. Respecto de la validez externa, el procedimiento se reviso y refino antes de ponerlo en práctica. Por otra parte los resultados obtenidos en cada una de estas fases se validaron con los creadores de la metodología, asegurando su corrección y validez.

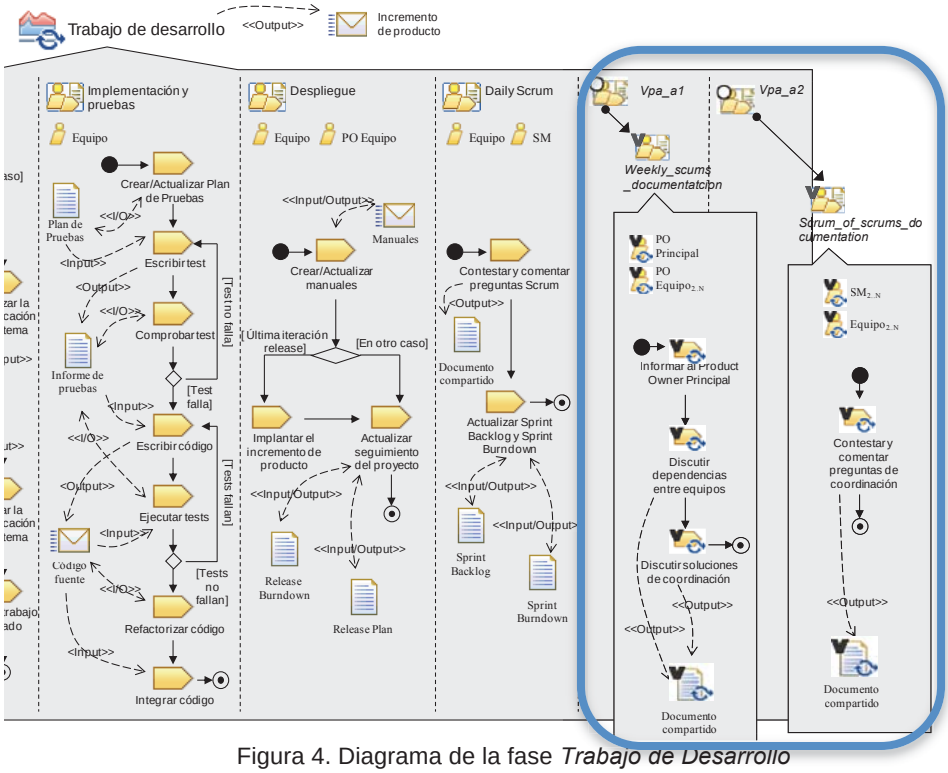


Figura 4. Diagrama de la fase *Trabajo de Desarrollo*

4 Estado del Arte

El interés de la comunidad científica por la adaptación de procesos empezó a adquirir mayor interés a partir de los trabajos de Sutton y Osterweil sobre programación de procesos software [17]. Partiendo de estas ideas, y ante la diversidad de organizaciones, cada vez aparecen más trabajos que abordan este problema, tales como los de Simidchieva et al. [2], y Barreto et al. [18]. Sin embargo, estos se enfocan unicamente en la adaptación de procesos. El empuje definitivo al campo de adaptabilidad de procesos software lo dieron Rombach [19] y Sutton [20], respectivamente al describir las similitudes de este problema con la adaptabilidad de productos software y la gestión aspectos transversales en el código, respectivamente, y plantear el uso de los enfoques de líneas de producto y desarrollo orientado a aspectos para adaptar los procesos.

En base a estos trabajos aparecieron propuestas como Puhlman et al [21], que proponen adaptar los procesos a partir de las líneas de producto, usando estereotipos [22], e incluso se crearon nuevas notaciones [8, 9]. Además las líneas de producto han incorporado la gestión de Rationale [14] como soporte a la decisión sobre variaciones [23], mientras que otros trabajos, como Ocampo et al. [24] y Nwoka et al. [25] proponen utilizarla para guiar en la evolución y mejora de los procesos software.

De acuerdo con Kulesza et al. [26], los aspectos se pueden utilizar para introducir flexibilidad en un sistema.. Las técnicas para aplicar gestión de aspectos en estructuras dinámicas se muestran en Laddaga et al. [27], y Apel et al. [28] que describe cómo implantar AOSE para modelar variabilidad en los productos software, mientras que Heo et al. [29], y Anastasopoulos et al. [30] muestran un mapeo de los elementos de AOSE y del enfoque de líneas de producto.

En [7]una revisión sistémica sobre cómo se adaptan los modelos de procesos en la literatura. En ella destaca que la adaptación de procesos se lleva a cabo por una parte en base a variaciones bien acotadas que afectan a pocos elementos, y por otra, grandes variaciones que son capaces de modificar diversos elementos del proceso a la vez. De forma similar se concluye que no existe un consenso acerca de la notación utilizada a la hora de adaptar procesos. Con el fin de resolver las necesidades y principales limitaciones encontradas, se propone el entorno que se presenta en el siguiente apartado.

Tras un análisis de los trabajos encontrados en la literatura se llega a la conclusión de que la comunidad científica está enfocada actualmente en proporcionar solución a la adaptación de procesos software, tal y como la industria reclama. Sin embargo no existen iniciativas que aborden de forma sistemática la adaptación y la mejora de procesos en base al conocimiento de la adaptación

5 Conclusiones

Las organizaciones dedicadas al desarrollo de software son cada día más conscientes de la necesidad de implantar modelos de procesos en proyectos. Sin embargo, si solamente se limitan a implantarlos pero no hacen nada porque el modelo de procesos case con sus características, antes o después tenderán a abandonarlo, y a perder la oportunidad de mejora y el esfuerzo (tiempo y dinero) invertido.
En este artículo se ha presentado un entorno para la institucionalización de procesos software y el paradigma de procesos ricos en variantes. Ambos tienen como objetivo dar soporte a las organizaciones en la tarea de asimilar los procesos software hasta llegar al punto en que éstos se integren como activos de la propia organización. Tal y como muestran los resultados del caso de estudio que se ha descrito, los mecanismos de variabilidad que incluye el paradigma, permiten adaptar los procesos reduciendo esfuerzo, complejidad y asegurando la consistencia de la adaptación.
Por otra parte, dentro del caso de estudio se ha "construido" un proceso rico en variantes que actualmente está en ejecución (fase 2 del ciclo de institucionalización), una vez que ésta termine será posible completar un caso de estudio para validar el entorno completo.

6 Trabajo Futuro: Hacia las Líneas de Contexto

El trabajo presentado en este artículo se está materializando en una herramienta de soporte a la inclusión de variabilidad en procesos software y facilitar su adaptación. Como trabajo futuro se plantea finalizar la herramienta que dé soporte al entorno, así

como la realización de más casos de estudio para validar entorno. Por otra parte está planeado iniciar la investigación en líneas de contexto.

A partir de los casos de estudio realizados se ha detectado la necesidad de extender la propuesta para gestionar la variabilidad desde el nivel de contexto y establecer su trazabilidad con el nivel de variabilidad en el proceso. En la introducción se ha descrito la necesidad que tienen los procesos de adaptarse al contexto en el que se van a implantar. Todos los seres vivos han evolucionado y se han diversificado en base a aplicar esta regla cientos y cientos de veces. Sin embargo, si tomamos como base la práctica que se utiliza actualmente en Ingeniería Agrícola, a la hora de plantar un árbol, no se elige uno al azar y se espera que unos cuantos de miles de generaciones después evolucione y con ello se adapte mejor al contexto en el que se halla.

Esta es la solución que en parte se plantea con la institucionalización de procesos. Se está permitiendo que los procesos, por genéricos que sean, acaben adaptándose y encajando en el contexto. Pero encajarán mucho antes si el proceso con el que se inicia la institucionalización es adecuado para ese contexto. En este caso el entorno sólo deberá perfilar el proceso final, en lugar de moldearlo totalmente.

Tomando como base las prácticas en Ingeniería Agrícola, por una parte los tratados en botánica determinan que tipo de terrenos son los más apropiados para cada árbol. A la hora de realizar una plantación, en primer lugar se hace un análisis del terreno y se estima cuanto se parece éste al terreno idóneo. Finalmente, el árbol elegido será aquel cuyo terreno es más similar.

Traspasando este enfoque a los procesos software se hace necesario no solo evaluar la variabilidad y adaptabilidad de los procesos, si no que tipos de contextos son los más idóneos para implementarlo. Todos los procesos, incluso los estándares, están diseñados para ejecutarse en un contexto (aunque sea irreal en el caso de los estándares). Acotando cuanto se parece un contexto c_i dado respecto del contexto C, se pueden determinar las variaciones a realizar sobre el proceso P para convertirlo en el proceso p_i que engrane perfectamente en el contexto anterior.

Un enfoque similar al descrito en este trabajo sobre los procesos ricos en variantes, debe plantearse sobre los contextos. No existen dos contextos iguales, pero si serán muy parecidos entre sí, con lo que pueden tratarse como una familia, en la que sus miembros quedan definidos a través de sus similitudes y diferencias. Y usar éstas como la base sobre la que facilitar la implantación de procesos software.

Para determinar el grado de similitud entre dos contextos, en primer lugar deben determinarse los factores que los definen y acotar cómo puede variar esos factores. En base a esto, la similitud entre dos contextos se puede simplificar en cuanto a las variaciones que estos factores sufren desde el proceso inicial p al proceso p_i. Y estas variaciones en el contexto implicarán variaciones en el proceso rico en variantes.

7 Agradecimientos

Este trabajo ha sido financiado por el programa FPU del Ministerio de Educación, y los proyectos ORIGIN (CDTI y FEDER IDI-2010043(1-5)), PEGASO/MAGO (MICINN and FEDER, TIN2009-13718-C02-01), y ALTAMIRA (JCCM, Fondo Social Europeo, PII2I09-0106-2463).

8 Referencias

1. Fuggetta, A., *Software process: a roadmap*, in *The Future of Software Engineering*, A. Finkelstein, Editor 2000, ACM: Limerick, Ireland. p. 25-34.
2. Simidchieva, B.I., L.A. Clarke, and L. Osterweil. *Representing Process Variation with a Process Family*. in *ICSP 2007*. 2007. Minneapolis, USA: Springer-Verlag.
3. Denger, C., F. Elberzhager, and T. Schulz, *Customization Approach for Inspection considering Influence and Variation Factors*, in *BMF Project*, F. IESE, Editor 2008, Fraunhoufer IESE: Kaiserslautern.
4. Chrissis, M.B., M. Konrad, and S. Shrum, *CMMI: guidelines for process integration and product improvement*. The SEI series in software engineering. Vol. 1. 2006, Boston: Pearson. 676.
5. Arbaoui, S., et al. *Languages and Mechanisms for Software Processes and Manufacturing Enterprise Processes: Similarities and Differences*. in *ICEIS*. 2003.
6. McFeeley, R., *IDEAL: A Users Guide for Software Process Improvement, Handbook CMU/SEI-96-HB-001*, 1996, Software Engineering Institute, Carnegie Mellon University: Pittsburgh, USA.
7. Martínez-Ruiz, T., et al., *Requirements and Constructors for Modeling Variability in Software Processes, a Systematic Review* Software Quality Journal, 2012. **20**(1): p. 229-260.
8. Martínez-Ruiz, T., F. García, and M. Piattini, *Towards a SPEM v2.0 Extension to Define Process Lines Variability Mechanisms*, in *SERA*, R. Lee, Editor 2008, Springer Verlag: Praga. p. 115-130.
9. Martínez-Ruiz, T., F. García, and M. Piattini, *Enhanced Variability Mechanisms to Manage Software Process Lines*, in *EUROSPI 2009*2009, Publizon: Alcalá de Henares (Madrid). p. 12.13-12.23.
10. Clements, P. and L. Northrop, *Software Product Lines. Practices and Patterns*2002, Boston: Addison-Wesley.
11. Martínez-Ruiz, T., et al., *Applying AOSE Concepts for Modeling Crosscutting Variability in Variant-Rich Processes*, in *37th SEAA*, S. Biffl, et al., Editors. 2011, IEEE: Oulu, Finlandia. p. 334-338.
12. Filman, R.E., et al., *Aspect-Oriented Software Development*2004, Boston, MA: Addison-Wesley.
13. Martínez-Ruiz, T., F. García, and M. Piattini, *Managing Process Diversity by Applying Rationale Management in Variant Rich Processes*, in *PROFES 2011*, D. Caivano, et al., Editors. 2011, Springer Verlag: Torre Cane, Italy. p. 128-142.
14. Dutoit, A.H., et al., *Rationale Management in Software Engineering: Concepts and Techniques*, in *Rationale Management in Software Engineering*, A.H. Dutoit, et al., Editors. 2006, Springer: Heidelberg. p. 1-45.
15. OMG, *Software Process Engineering Metamodel Specification*, 2007, Object Management Group.

16. Brereton, P., B. Kitchenham, and D. Budgen, *Using a Protocol Template for Case Study Planning*, in *Proceedings of EASE 2008*2008, BCS-eWiC.
17. Sutton, S. and L.J. Osterweil. *PDP: Programming a Programmable Design Process*. in *8th Int. Workshop on Software Specification and Desing*. 1996.
18. Silva Barreto, A., L. Murta, and A.R. Rocha. *Software Process Definition: a Reuse-Based Approach*. in *XXXIV Conferencia Latinoamericana de Informática*. 2008. Santa Fe, Argentina.
19. Rombach, D., *Integrated Software Process and Product Lines*, in *ISPW*, M. Li, B. Boehm, and L. Osterweil, Editors. 2005, Springer: Beijing, China. p. 83-90.
20. Sutton, S.M., *Aspect-Oriented Software Development and Software Process*, in *ISPW*, M. Li, B. Boehm, and L.J. Osterweil, Editors. 2005, Springer. p. 177-191.
21. Puhlmann, F., et al., *Process Family Engineering: Variability Mechanisms for Process Models*, in *PESOA Report*2005, PESOA Project: Postdam, Alemania.
22. Schneiders, A. and M. Weske, *Activity Diagram Based Process Family Architectures for Enterprise Application Families*, in *Enterprise Interoperability: New Challenges and Approaches*, G. Doumeingts, et al., Editors. 2007, Springer: London. p. 67-76.
23. Dutoit, A.H., et al., *Rationale Management in Software Engineering*2006, Heidelberg: Springer.
24. Ocampo, A. and J. Münch, *Rationale Modeling for Software Process Evolution.* Software Process Improvement and Practice, 2008. **14**(2): p. 85-105.
25. Nkwoca, A., J. Hall, and L. Raspanotti, *Design Rationale Capture for Process Improvement int he Globalised Enterprise: An Industrial Study*, in *Faculty of Mathematics and Computing*2010, The Open University: Milton Keynes.
26. Kulesza, U., et al., *Improving Extensibility of Object-Oriented Frameworks with Aspect-Oriented Programming*, in *ICSR*, M. Morisio, Editor 2006, Springer-Verlag: Torino, Italia. p. 231-245.
27. Laddaga, R., P. Robertson, and H. Shrobe, *Aspects of the real world.* OOPSLA, 2001.
28. Apel, S., et al., eds. *Combining Feature-Oriented and Aspect-Oriented Programming to Support Software Evolution*. In AMSE'05, at ECOOP'05, ed. W. Cazzola, et al.2005, Fakultät für Informatik, Universität Magdeburg. 3-16.
29. Heo, S.-H. and E.M. Choy, *Representation of Variability in Software Product Line Using Aspect-Oriented Programming*, in *4th SERA*, Y.-T. Song, Editor 2006, IEEE CS: Washington DC, USA. p. 66-73.
30. Anastasopoulos, M. and D. Muthig, *An Evaluation of Aspect-Oriented Programming as a Product Line Implementation Technology*, in *8th ICSR*2004, Springer-Verlag: Madrid. p. 141-156.

Un experimento para validar transformaciones QVT para la generación de modelos de servicios en SoaML desde modelos de procesos de negocio en BPMN2

Andrea Delgado[1], Francisco Ruiz[2], Ignacio García-Rodríguez de Guzmán[2], Mario Piattini[2]

[1] Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Julio Herrera y Reissig 565, 11300, Montevideo, Uruguay
adelgado@fing.edu.uy
[2]Instituto de Tecnologías y Sistemas de Información, Universidad de Castilla – La Mancha, Camino de Moledores s/n, 13051, Ciudad Real, España
{francisco.ruizg, ignacio.grodriguez, mario.piattini}@uclm.es

Abstract. La realización de procesos de negocio (PNs) mediante servicios presenta varias ventajas frente a otras opciones tales como desacoplar la definición de los PN de las tecnologías que los implementan, promover la reutilización de los servicios entre distintos PNs, y facilitar el análisis del impacto de los cambios, tanto en la definición de los PNs como en su implementación. En el framework MINERVA se propone un enfoque MDA para la generación automática de servicios en SoaML desde PNs en BPMN2, mediante transformaciones QVT. Hemos validado las transformaciones propuestas por medio de un experimento que se centró en evaluar dos características de calidad: la adecuación de las transformaciones propuestas (en relación con lo que los usuarios esperan modelar por sí mismos a partir del modelo de PN) y la entendibilidad de los modelos de servicios que se generan (por medio del significado de los elementos generados y sus relaciones). Hemos encontrado que el 82% y el 75% de los participantes prefiere y entiende, respectivamente, el diseño que proponemos.

Keywords: validación empírica, experimentación, generación automática, transformaciones QVT, SoaML y BPMN2, modelos de PNs y servicios.

1 Introducción

La realización de procesos de negocio (PNs) mediante servicios (Service Oriented Computing, SOC) [1] permite desacoplar la definición de los PNs de las tecnologías que los implementan, soportando la visión horizontal de la organización basada en PNs. La implementación con servicios presenta, entre varias ventajas frente a otras opciones, la reutilización de servicios entre distintos PNs, facilitar el análisis del impacto de los cambios tanto en la definición de los PNs como en su implementación, con mínimo impacto de unos en otros, permitiendo la introducción de cambios y nuevos requisitos de forma más ágil [2]. La derivación automática de servicios desde

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 227-240, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

PNs mediante desarrollo dirigido por modelos (Model Driven Development, MDD) [3] permite además, entre otras cosas, explicitar la trazabilidad de las relaciones entre los elementos de los modelos de PNs y servicios, promoviendo también la reutilización del conocimiento que se encuentra embebido en las transformaciones definidas entre los distintos metamodelos y modelos. El modelado de PNs y servicios es un aspecto clave para soportar la visión horizontal de la organización basada en PNs, su gestión, optimización y mejora (Business Process Management, BPM) [4] [5] [6]. El framework MINERVA [7] que hemos definido soporta esta visión horizontal con foco en la mejora continua de PNs realizados mediante servicios con desarrollo dirigido por modelos. Entre otros elementos, incluye una metodología nombrada BPSOM [8] y transformaciones con el lenguaje Query/View/Transformations (QVT) [9], para guiar el desarrollo de servicios desde PNs [10], proveyendo soporte a la trazabilidad entre PNs y servicios, y a la introducción de mejoras ágilmente en ambos.

El objetivo de este artículo es presentar la validación de las transformaciones QVT integradas en MINERVA para la generación automática de modelos de servicios especificados en Service Oriented Architecture Modeling Language (SoaML) [11] desde modelos de PNs especificados en Business Process Model and Notation (BPMN2) [12] que hemos realizado mediante un experimento, no así el detalle de las transformaciones que pueden verse en [10]. Como principal resultado hemos obtenido que la generación automática provee un diseño de servicios adecuado y entendible, para ser utilizado en el proceso de desarrollo de software para el desarrollo de servicios. El resto del documento está organizado de la siguiente forma: en la sección 2 se presentan brevemente las transformaciones QVT y la correspondencia entre elementos de los metamodelos BPMN2 y SoaML definidas, en la sección 3 se presenta el detalle de la definición del experimento y el análisis de resultados obtenidos, en la sección 4 se presentan trabajos relacionados y finalmente en la sección 5 se elaboran algunas conclusiones y el trabajo futuro.

2 Transformaciones QVT de BPMN2 a SoaML

Para generar los modelos de servicio en SoaML desde modelos de PNs en BPMN2 definimos en primer lugar la correspondencia entre elementos en los metamodelos de BPMN2 y SoaML apoyados en el razonamiento realizado para la construcción de una ontología de modelado de PNs y servicios [13], en base a la evaluación de diversos estándares, metamodelos y notaciones existentes. En cuanto al modelo BPMN2, la principal construcción que identificamos para generar los servicios en SoaML corresponde al patrón de flujo de mensaje entre dos pools distintos, *Messageflow*, donde la actividad objetivo es una *ServiceTask* y la actividad origen una *Task*. La actividad implementada como servicio provisto (rol *provider*) es la identificada como *ServiceTask*, y la actividad *Task* que lo invoca tiene el rol *consumer*.

La generación automática solo requiere que el Arquitecto marque las acctividades desde las cuales generar los servicios del modelo como *ServiceTasks*, generando con los mismos nombres. Cuando hay otros elementos en el modelo BPMN2, como ser las *Interfaces*, *Operations* y *Messages* para el servicio provisto, mantenemos los nombres y definiciones asociándolos con el servicio generado para el proveedor. En cuanto al

modelo SoaML generado, puede tener tres formas distintas según el patrón de comunicaciones aplicado: bidireccional y unidireccional con Interfaces simples UML, o bidireccional con Interface SoaML ServiceInterface. Combinando las opciones de generación teniendo en cuenta los distintos casos en el modelo BPMN2 y el SoaML definimos seis transformaciones QVT, cada una de las cuales se compone de siete relaciones "top relation" y otras que son invocadas desde éstas. En todos los casos se generan los cinco diagramas que se muestran en la Fig. 1 obteniendo modelos de servicio SoaML completos desde el punto de vista de la especificación de los mismos, donde el usuario únicamente tiene que incluir la lógica del negocio al implementarlos.

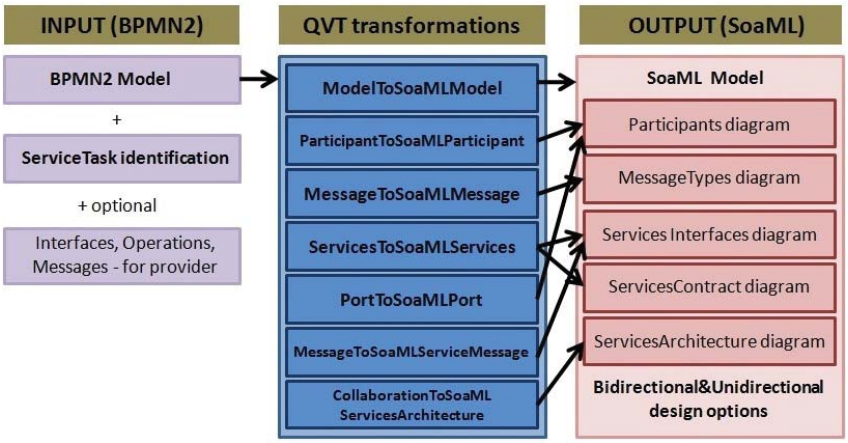


Fig. 1. Definición general de las transformaciones QVT para generar servicios

Como se puede ver en la Fig. 1, desde el BPMN2 *Model* generamos el SoaML *Model*, luego para cada *Pool* en el PN colaborativo generamos un *Participant* en el modelo de servicios, y para cada extremo de los *Messageflow* generamos un *MessageType*. Para cada par de actividades que cumplen el patrón de servicios presentado (.i. e. *Messageflow* entre dos *Pools* distintos con target una *ServiceTask* y source una *Task*) generamos la especificación completa de servicios para el patrón de comunicación elegido, incluyendo las *Interfaces* con *Operations* y *Parameters*, y el *ServiceContract* asociado con la definición de roles. Luego de generar los servicios, generamos los *Ports* y los asociamos a los *Participants* según los servicios provistos y requeridos por cada uno (*Service* and *Request Ports* respectivamente), y asociamos los *MessageType* generados como tipo de los parámetros de las operaciones. El PN colaborativo completo se corresponde con la *ServicesArchitecture* que incluye los *Participants, ServiceContracts* y el rol que cada participante juega en cada servicio.

3 Validación empírica de las transformaciones QVT

El experimento realizado tuvo como objetivo validar la sub-característica Adecuación (Suitability) de la característica Funcionalidad (Functionality), y la sub-

característica Entendibilidad (Understandability) de la característica Usabilidad (Usability), según las definiciones de ISO 9126 [14], de las transformaciones QVT definidas y de los modelos de servicios en SoaML generados mediante las transformaciones desde los modelos BPMN2. Fue realizado gracias a la participación de varios Ingenieros en Informática y Computación de la Universidad de Castilla – La Mancha, España y de la Universidad de la República, Uruguay, quienes evaluaron el diseño de servicios que proponemos. El experimento fue realizado siguiendo la definición, procedimientos y guías en [15] [16] [17], en base al proceso de seis etapas de [15]: definición, planificación, operación, análisis e interpretación, evaluación de la validez, y presentación y diseminación, que se describen a continuación.

3.1 Definición

El objetivo del experimento era evaluar si las transformaciones QVT que definimos provee a los diseñadores de software con un diseño de servicios en SoaML que se corresponde con lo que esperarían para realizar PNs con servicios y que pueden ser utilizados para el desarrollo de servicios. En términos de la plantilla GQM (Goal, Question, Metrics) [18] fue definido según la Tabla 1. La pregunta de investigación que planteamos es la siguiente:

¿Las transformaciones QVT definidas entre modelos BPMN2 y SoaML proveen a los Ingenieros de software con modelos de servicios que son adecuados a lo que esperarían modelar ellos mismos, así como usables como diseño de servicios en su desarrollo de servicios desde modelos de PNs?

Tabla 1. Definición del experimento en GQM.

Analizar	transformaciones QVT entre modelos BPMN2 y SoaML
con el propósito de	evaluar
con respecto a	la funcionalidad de las transformaciones y la usabilidad de los modelos SoaML generados
desde el punto de vista de	Ingenieros de software
en el contexto de	diseñar servicios para realizar Procesos de Negocio (PNs)

3.2 Planificación

En la etapa de planificación se prepara la realización del experimento según la definición previa. En la Fig. 3 se muestra el resumen del plan experimental, cuyos elementos se describen en esta sección.

Selección de contexto y sujetos. La realización del experimento requiere que los sujetos participantes tengan estudios de por lo menos cinco años en Informática con conocimientos generales de modelado y diseño de software, y notaciones para especificar dichos modelos, como puede ser UML.

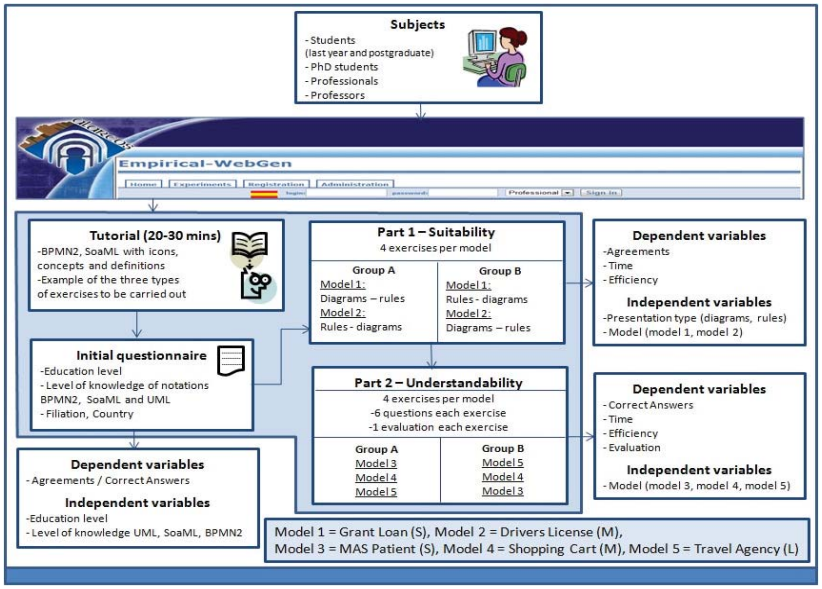


Fig. 3. Resumen del plan experimental

Con este objetivo, los participantes debían ser estudiantes del último año de una carrera de computación de cinco años, estudiantes de posgrado (Maestría y Doctorado), profesores de universidad, ingenieros profesionales egresados de universidad. Siendo que el grupo Alarcos de la Universidad de Castilla – La Mancha cuenta con una aplicación desarrollada específicamente para realizar experimentos online, Empirical-WebGen [19], decidimos realizar el experimento vía web, enviando email a varias personas que cumplían los requisitos para realizar el experimento, de los que veintiuno aceptaron.

Los modelos de PN usados en los ejercicios fueron seleccionados de casos reales y ejemplos conocidos, adaptados a las necesidades del experimento. Los cinco modelos de PN utilizados fueron: Otorgar préstamo de un banco, Cirugía Mayor Ambulatoria (CMA) de un hospital, Licencia de conducir de una oficina de Tránsito, Carrito de Compras del sitio Web de una Compañía de Ventas, y Reserva de viaje de una Agencia de viajes. Cada modelo de PN tenía una complejidad distinta asociada, que definimos según la cantidad de servicios que serían generados a partir del mismo, siendo S = pequeño (Small, cuatro servicios), M = medio (Medium, siete servicios) y L = grande (Large, once servicios).

Selección de variables. Las características a ser evaluadas para alcanzar los objetivos del experimento fueron seleccionadas según las definiciones en ISO 9126 [14] (ahora ISO/IEC 25000 [20]). Las variables dependientes del experimento corresponden a las sub-características Adecuación de Funcionalidad, y Entendibilidad de Usabilidad, que fueron medidas con las variables que se describen a continuación. Para la Adecuación, en primer lugar contamos los acuerdos con la opción correspondiente a

nuestra propuesta de diseño para derivar servicios desde modelos PN con las transformaciones, el tiempo que lleva a cada sujeto responder cada ejercicio, y la eficiencia en realizar cada ejercicio, calculada como cantidad de acuerdos/tiempo incurrido. Para la Entendibilidad, en primer lugar contamos las respuestas correctas en cada una de las seis preguntas en cada ejercicio sobre el diseño de servicios generado, así como el tiempo y la eficiencia igual que antes, pero utilizando la variable respuestas correctas en lugar de acuerdos, y finalmente pedimos que se evaluara la complejidad del diagrama SoaML generado en cada ejercicio.

Formulación de Hipótesis. Las respuestas a la pregunta de investigación para las sub-características de Adecuación y Entendibilidad son provistas directamente por el porcentaje de acuerdos y respuestas correctas para cada parte del experimento. Por lo tanto, no podíamos esperar resultados del tipo "utilizar esto es mejor que utilizar lo otro"; por lo cual las hipótesis que definimos están relacionadas con la evaluación de las amenazas a la validez del experimento según la influencia de la variación de las variables independientes en los resultados de las dependientes. Para la Adecuación definimos dos variables independientes: la presentación de las transformaciones QVT como diagramas SoaML generados o reglas textuales de correspondencia, y la complejidad de los modelos de PN; y para la Entendibilidad solo ésta última, las cuales se muestran en la Tabla 2.

Tabla 2. Hipótesis centrales para la evaluación de la Adecuación y Entendibilidad

Variable dependiente	Medida por	Hipótesis	Variables independientes
Adecuación	Acuerdos Tiempo Eficiencia	**H0.a** = la presentación de las transformaciones QVT como diagramas o reglas de correspondencia textuales no tiene efecto en los resultados de Adecuación **H1.a = ¬ H0.a**	Tipo de presentación (diagrama, reglas textuales)
		H0.b = la Complejidad del modelo de PN no tiene efecto en la Adecuación de las transformaciones QVT **H1.b = ¬ H0.b**	Complejidad del modelo de PN
Entendibilidad	Respuestas correctas Tiempo Eficiencia Evaluación	**H0.c** = la Complejidad del modelo de PN no tiene efecto en la Entendibilidad del resultado de las transformaciones QVT **H1.c = ¬ H0.c**	Complejidad del modelo de PN

Definimos también hipótesis complementarias para evaluar las implicancias del nivel educativo y el conocimiento de los lenguajes de notación UML, SoaML y BPMN2 pedidos en el cuestionario inicial en el sitio web (durante la realización del experimento), que no presentamos aquí por limitaciones de espacio y a que los resultados sobre las correlaciones no fueron concluyentes.

Diseño del experimento. El experimento fue definido en dos partes: la primera correspondiente a la variable dependiente Adecuación y la segunda a la Entendibilidad. Los sujetos fueron repartidos al azar en dos grupos, Grupo A y Grupo

B, mediante la asignación de cada sujeto que aceptaba realizar el experimento a un grupo distinto en forma secuencial al recibir el email de respuesta, comenzando con el Grupo A, de forma que la asignación fuera balanceada. De esta forma, once sujetos fueron asignados al Grupo A y diez al Grupo B.

Parte 1 - Adecuación. Se utilizaron dos modelos de PN: Otorgar préstamo correspondiendo al Modelo 1, y Licencia de Conducir correspondiendo al Modelo 2; los mismos diagramas y reglas textuales de correspondencia fueron entregados a cada Grupo, pero en orden distinto. El Grupo A realizó la primer asignación de tareas sobre el Modelo1 respondiendo para cada ejercicio de tipo de diagramas SoaML, primero las preguntas con tipo de presentación diagramas y segundo las reglas textuales de correspondencia; la segunda asignación de tareas se realizó sobre el Modelo 2 respondiendo para cada ejercicio primero las preguntas con tipo de presentación reglas textuales y segundo diagramas. El Grupo B lo hizo en forma opuesta, para el Modelo 1 primero las preguntas con tipo de presentación reglas textuales y segundo diagramas, y para el Modelo 2 primero las preguntas con tipo de presentación diagramas y segundo las de reglas textuales. La complejidad de cada modelo de PN corresponde para el Modelo1 a pequeña (cuatro servicios) y para el Modelo 2 a media (siete servicios). El diseño del experimento Parte 1 – Adecuación corresponde a un 2x2 factorial y se muestra en la Tabla 3.

Tabla 3. Diseño del experimento Parte 1 - Adecuación

	Modelo de PN	
Adecuación	Otorgar Préstamo	Licencia de conducir
Diagramas-Reglas	Grupo A	Grupo B
Reglas-Diagramas	Grupo B	Grupo A

Para cada modelo de PN se proponían cuatro ejercicios correspondientes a cuatro diagramas SoaML generados por las transformaciones, y en cada uno se brindaban cuatro diseños distintos tanto como diagramas SoaML como reglas textuales, de los cuales los sujetos solo podían elegir una opción. Las opciones de diseño corresponden a: nuestra generación que identifica todas las posibles ServiceTasks (según semántica de las actividades en el PN) entre dos Pools para automatizar la generación de servicios lo más posible; combinación en un único servicio de dos ServiceTasks relacionadas conceptualmente pero en dos Messageflows distintos; un subconjunto de posibles ServiceTasks para automatizar la generación de servicios, o todas las actividades involucradas en messageFlows entre distintos pools identificadas como ServiceTasks sin importar su semántica. Para cada diagrama SoaML generados en evaluación (ServicesArchitecture, Service Interfaces, ServiceContract y Participants&Services) las cuatro opciones de diseño se mantienen entre los diagramas, conformando así el diseño completo de servicios (el orden de presentación de las opciones se cambia para cada diagrama). En la Fig. 4 se presenta un ejemplo para el diagrama de ServiceArchitecture y dos opciones de diseño de las cuatro provistas.

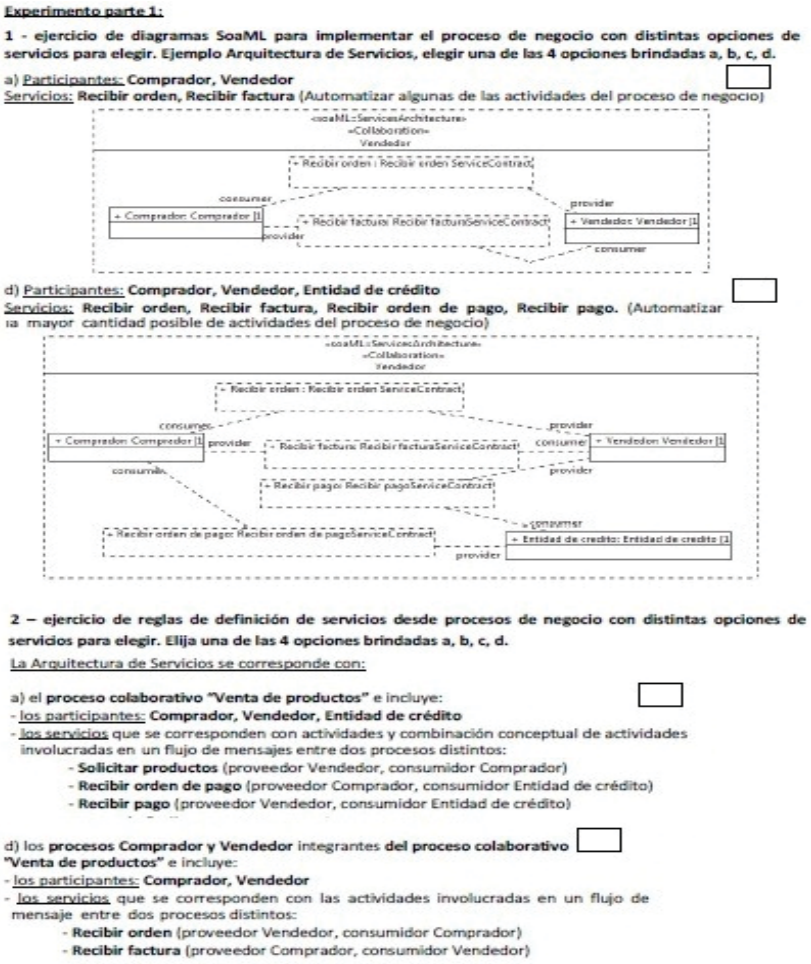


Fig. 4. Parte 1 ejemplo de ejercicio de diagramas y reglas textuales para Adecuación

Parte 2 - Entendibilidad. Para esta parte se utilizaron tres modelos de PN adicionales: la Cirugía Mayor Ambulatoria (CMA) correspondiente al Modelo 3, el Carrito de Compras del sitio Web correspondiente al Modelo 4 y la Reserva de viaje correspondiente al Modelo 5. La complejidad de los modelos es: pequeña para el Modelo 3 (cuatro servicios), media para el Modelo 4 (siete servicios) y grande para el Modelo 5 (once servicios). Igual que en la parte anterior, para cada modelo de PN se proponían cuatro ejercicios correspondientes a cuatro diagramas SoaML generados por las transformaciones (pero solamente con la opción de diseño de nuestra generación), y en cada diagrama seis preguntas sobre el significado de los elementos en el mismo, para ser respondidas con Verdadero o Falso. El orden de las seis preguntas era al azar para cada individuo y para cada ejercicio.

Para cada ejercicio sobre un diagrama SoaML se pedía también la evaluación de la complejidad del modelo en una escala de 1 a 5 (1 más simple, 5 más complejo). En la Fig. 5 se presenta un ejemplo para el diagrama de ServicesArchitecture y las seis preguntas relacionadas, así como la evaluación de la complejidad.

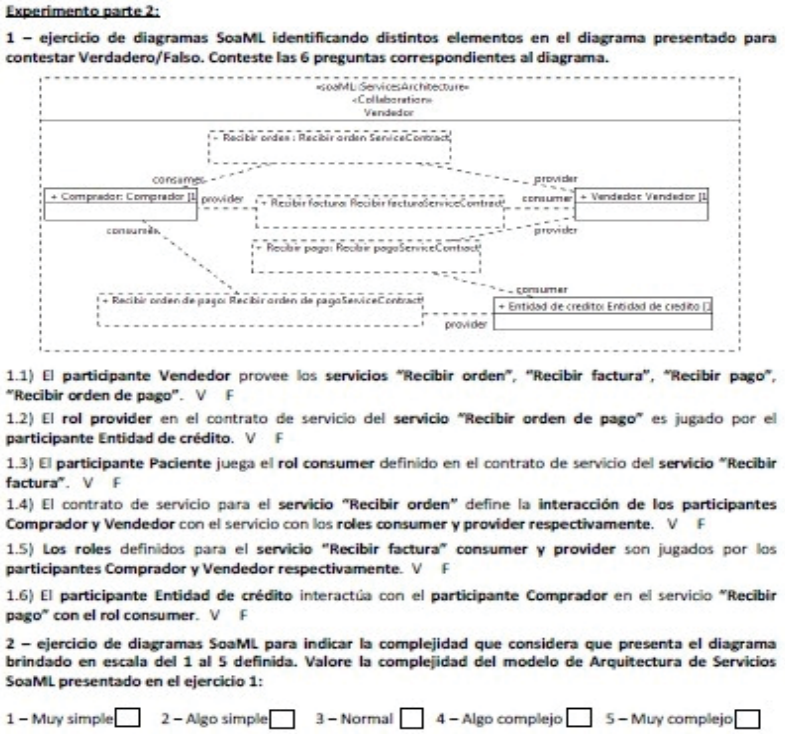


Fig. 5. Parte 2 ejemplo de ejercicio de diagramas SoaML para Entendibilidad

Materiales experimentales. Tres materiales distintos fueron entregados a ambos grupos para realizar el experimento: un tutorial en las dos notaciones a utilizar, BPMN2 y SoaML, y un ejemplo para cada tipo de ejercicio que tenían que realizar.

Evaluación de la validez. Las amenazas a la validez del experimento fueron analizadas como parte de la planificación del experimento, y se describen debajo:
- *validez de construcción*: las medidas seleccionadas de acuerdos/respuestas correctas, tiempo y eficiencia son normalmente utilizadas en Ingeniería de Software empírica para medir variables dependientes. Que el tiempo registrado pueda no ser el real al ser el experimento online se redujo con un time-out y avisando a los sujetos que una vez comenzado el experimento se debía terminar sin interrupciones.
- *validez interna*: los modelos de PN y dominios son conocidos de la vida diaria, de forma de mitigar la necesidad de comprender el problema. Para reducir los efectos de

fatiga cada parte duraba una hora y se podían realizar en cualquier momento. Los sujetos aceptaron realizar el experimento sin ser forzados, explicando la importancia del mismo para el trabajo de investigación.
- *validez externa*: Los sujetos fueron seleccionados según las restricciones definidas contando con una muestra heterogénea de estudiantes de doctorado, de posgrado y grado (último año), profesores de universidad y profesionales.
- *validez de conclusiones*: El conjunto completo de datos consistió en 336 respuestas para la parte 1, 1.512 para la parte 2 y 252 respuestas para la parte 2 de evaluación de la complejidad, correspondientes a los 21 sujetos que realizaron el experimento.

3.3 Operación del experimento

Preparación. El experimento fue realizado por parte de un Doctor Ingeniero con conocimiento de modelado, para evaluar los tiempos y corregir errores en el material.

Ejecución. Los sujetos que aceptaron recibieron vía email la descripción de las partes del experimento, su duración, y secuencia de realización (parte 1 antes que parte 2, aún en momentos distintos). Se otorgó una semana para hacer el experimento, y se enviaron los links para acceder a la aplicación WebGen y al tutorial, la asignación de grupo y el usuario y password. Las respuestas se almacenan en la base de datos de la aplicación, y pueden ser extraídas en formatos como Excel o pdf.

Validación de datos. La aplicación web tiene controles para asegurar que cada ejercicio y pregunta se respondiera según las definiciones, evitando la navegación hacia adelante y atrás entre ejercicios, asegurando que solo se avanza al siguiente una vez que se terminó el anterior. Cada parte del experimento fue chequeada por datos perdidos, que no fue el caso siendo las veintiún ejecuciones del experimento válidas.

3.4 Análisis de datos e interpretación de resultados

Para el procesamiento de los datos se utilizó el paquete estadístico SPSS. Como ya fue mencionado, las respuestas a la pregunta de investigación están dadas directamente por el porcentaje de acuerdos y respuestas correctas para cada parte del experimento, los que se presentan en la Tabla 4 a continuación.

Tabla 4. Resultados generales para las variables de Adecuación y Respuestas correctas

Variables dependientes	Medidas por	Media	Desv. estándar
Adecuación	Acuerdos (%)	**82,14**	38,36
	Tiempo (s)	154,07	120,61
	Eficiencia	0,764	0,348
Entendibilidad	Respuestas correctas (%)	**75,38**	12,56
	Tiempo (s)	290,83	85,92
	Eficiencia	0,277	0,081
	Evaluación	Media=3	0,472

El porcentaje de Acuerdos en Adecuación indica que la solución de diseño de servicios que proponemos con las transformaciones QVT es apropiado para el diseño de servicios que los usuarios esperan, en promedio en un 82% de los casos. Para la Entendibilidad de los modelos de servicios en SoaML generados, el porcentaje de Respuestas Correctas indica que los usuarios los entienden en promedio en un 75% de los casos. Estos porcentajes indican entonces, que la solución que proponemos para el diseño de servicios a obtener desde modelos de PN en forma automática, es adecuada y entendida por los Ingenieros de software en esos altos porcentajes. Con respecto al tiempo promedio para cada ejercicio, en la Parte 2 es el doble que en la Parte 1, con el porcentaje de respuestas correctas menor que el de acuerdos, con lo cual la eficiencia para la Parte 2 es la cuarta parte de la eficiencia de la Parte 1.

Los resultados presentados pueden descomponerse además para Adecuación según el tipo de presentación (diagrama, reglas textuales) y la complejidad del modelo de PN (Modelo 1 pequeña y Modelo 2 media), y para Entendibilidad según complejidad del modelo de PN (Modelo 3 pequeña, Modelo 4 media y Modelo 5 grande). Estos resultados se muestran en las Tabla 5 para Adecuación y Tabla 6 para Entendibilidad, y en la Fig. 6 como diagramas de caja para Acuerdos por Tipo de presentación y Complejidad del modelo, y Respuestas correctas por Complejidad del modelo.

Tabla 5. Resultados de Adecuación por tipo de presentación y modelo

Adecuación	Tipo de presentación				Complejidad del Modelo de PN			
	Diagramas		Reglas textuales		Modelo 1		Modelo 2	
	Media	Dev.std	Media	Dev.std.	Media	Dev.std.	Media	Dev.std.
Acuerdos (%)	**84,83**	13,01	**79,09**	11,30	**88,58**	31,8	**75,17**	43,1
Tiempo (s)	135,11	156,67	173,33	203,07	195,69	231,36	112,75	97,61
Eficiencia	0,832	0,374	0,625	0,279	0,698	0,387	0,798	0,293

Tabla 6. Resultados de Entendibilidad por modelo

Entendibilidad	Complejidad del Modelo de PN					
	Modelo 1		Modelo 2		Modelo 3	
	Media	Dev.std	Media	Dev.std.	Media	Dev.std.
Resps. Correctas (%)	**75,93**	12,45	**76,46**	13,03	**73,75**	13,76
Tiempo (s)	259,63	66,16	263,29	26,78	349,56	115,07
Eficiencia	0,308	0,091	0,293	0,058	0,229	0,076
Evaluación	Mediana=3	0,638	Mediana=3	0,685	Mediana=4	0,789

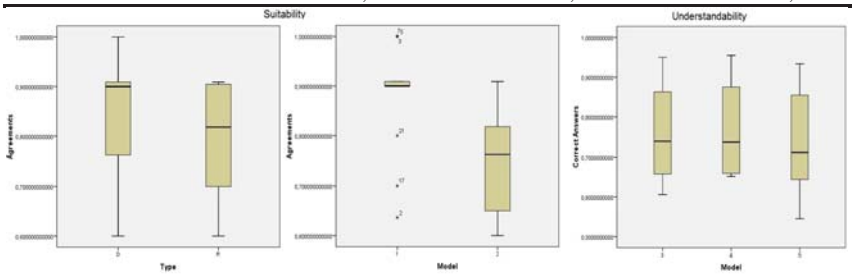


Fig. 6. Acuerdos por tipo de presentación& modelo, Respuestas correctas por modelo

Se puede observar que el porcentaje de Acuerdos es mayor para Diagramas (84,83%) que para Reglas (79,09%), los tiempos por Diagramas (135,11s) son menores que los de Reglas (173,33s), por lo que la eficiencia es mayor para los Diagramas (0,832) que para las Reglas (0,625). El porcentaje de Acuerdos es mayor para el Modelo 1 (88,58%) que para el Modelo 2 (75,17%), y los tiempos para el Modelo 1 (195,69s) también son mayores que para el Modelo 2 (112,75s), los resultados para la eficiencia son similares pero mayores para el Modelo 2 (0,798) que para el Modelo 1 (0,698). Los resultados para Entendibilidad son similares para los tres modelos en cuanto a Respuestas correctas: Modelo 1 (75,93%), Modelo 2 (76,46%) y Modelo 3 (73,75%), incrementándose los tiempos al incrementarse la complejidad y bajando la eficiencia. Luego del análisis de las estadísticas descriptivas, se realizó el contraste de hipótesis para evaluar si las diferencias son significativas estadísticamente. En la Tabla 7 se presentan los resultados obtenidos.

Tabla 7. Niveles de significancia para Tipo de presentación y Complejidad del modelo

		Tipo de presentación	Comp. Modelo de PN
		p-value	p-value
Adecuación	Acuerdos	0,150	**0,002**
	Tiempo	**0,048**	**0,000**
	Eficiencia	0,080	0,280
		Complejidad del Modelo de PN	
		Coeficiente Corr.	p-value
Entendibilidad	Resps. Correctas	-0,021	0,422
	Tiempo	0,436	**0,033**
	Eficiencia	-0,407	**0,049**
	Evaluación	0,279	**0,000**

Luego de determinar que la distribución de los datos obtenidos era normal (mediante el test de Kolmogorov-Smirnov) lo que permite utilizar tests paramétricos (requieren menos datos), se seleccionaron los tests más adecuados según los diseños experimentales de cada parte: para la Adecuación un test ANOVA con nivel de significación $\alpha = 0.05$, y para la Entendibilidad se utilizó el coeficiente de correlación de Pearson también con nivel de significación $\alpha = 0.05$. El nivel de significación indica la probabilidad de rechazar la hipótesis nula cuando es cierta (error de tipo I) indicando un nivel de confianza del 95%. Por más detalles de análisis estadístico en Ingeniería de Software se pueden consultar [15][16].

Como se puede ver en la Tabla 7, para la Adecuación se encontraron resultados significativos para la variable Acuerdos para Complejidad del modelo de PN, y para la variable Tiempo para el Tipo de presentación y Complejidad del modelo. En base a esto, la hipótesis nula H0.b para Acuerdos y Complejidad del modelo es rechazada, y la hipótesis nula H0.a para Tiempo y Tipo de presentación y Complejidad del modelo también. Se puede concluir que los sujetos acuerdan más con la solución generada para modelos pequeños, como era esperable ya que la Complejidad del modelo de PN influencia la visión general del diseño. El tipo de presentación en diagramas ayudó a los sujetos a responder en menor tiempo que las reglas textuales. La Complejidad del modelo influenció el tiempo de respuesta pero en dirección contraria a la esperada, ya que el Modelo 1 llevó más tiempo que el Modelo 2, que es más complejo. Esto puede

deberse a que el Modelo 1 fue el primero en ser contestado, por lo que para el Modelo 2 los sujetos se sentían más cómodos con los ejercicios y la aplicación web.

Para la Entendibilidad se encontraron resultados significativos para las variables Tiempo, Eficiencia y Evaluación del modelo SoaML para Complejidad del modelo de PN, por lo que la hipótesis nula H0.c es rechazada para cada una. Se puede concluir que la Complejidad de los modelos no afecta las Respuestas correctas, lo que fue visto en la estadística descriptiva donde los porcentajes eran similares para los tres. En el caso del Tiempo se puede concluir que a medida que la Complejidad del modelo aumenta, lleva más tiempo su comprensión por parte de los sujetos, según lo esperado. La eficiencia para cada modelo es inversa a los tiempos, siendo los porcentajes de Respuestas correctas similares, significando que a medida que la Complejidad del modelo crece, los sujetos son menos eficientes en comprenderlos. La evaluación de la complejidad de los modelos SoaML es la misma para modelos de PN de complejidad pequeña y media, y crece para modelos de PN de complejidad grande; a modelos de PN más complejos también lo son los modelos de servicios obtenidos.

4 Trabajos relacionados

En cuanto a la generación automática de servicios desde PNs, no existen en nuestro conocimiento, propuestas que como la nuestra, generen modelos completos en SoaML (todos los diagramas necesarios para especificar los servicios) desde modelos en BPMN2 con QVT. Algunas propuestas de automatización utilizan otros lenguajes de modelado y transformación de modelos (ej. ATL) pero no proveen como la nuestra un diseño completo de servicios desde PNs [21], y plantean diseños de servicios distintos. En experimentación en base a modelos, existen varios trabajos del grupo Alarcos [22][23] que plantean comparación de modelos con diferentes elementos, y modelado o reglas textuales, pero el diseño general de nuestro experimento es único ya que agrega como novedad la comparación de distintas opciones de diseño en el mismo ejercicio (parte 1) en lugar de con distintas ejecuciones de cada ejercicio [16].

5 Conclusiones y trabajo futuro

El experimento presentado tuvo como objetivo evaluar la Adecuación de las transformaciones QVT definidas para la generación automática de modelos de servicios en SoaML desde modelos de PN en BPMN2, y la Entendibilidad de los modelos SoaML generados. En base al análisis e interpretación de datos realizado sobre los resultados obtenidos, se puede observar que las transformaciones QVT definidas para la generación automática de modelos de servicios desde modelos de PN serían adecuadas y usables para el desarrollo automático de servicios desde PNs. Sin embargo, no es posible aún generalizar los resultados, lo que esperamos poder hacer luego de realizar varias réplicas del experimento previstas como trabajo futuro.

Agradecimientos. Este trabajo ha sido financiado parcialmente por la Agencia Nacional de Investigación e Innovación (ANII, Uruguay), proyecto ALTAMIRA (Junta de Comunidades de

Castilla-La Mancha, España, Fondo Social Europeo, PII2I09-0106-2463), proyecto PEGASO/MAGO (Ministerio de Ciencia e Innovación MICINN, España, Fondo Europeo de Desarrollo Regional FEDER, TIN2009-13718-C02-01), proyecto INGENIOSO (Junta de Comunidades de Castilla-La Mancha, España, PEII11-0025-9533) y proyecto MOTERO (Junta de Comunidades de Castilla-La Mancha, España, PEII11-0366-9449).

Referencias

[1]Papazoglou M, Traverso P, Dustdar S, Leymann F. Service-Oriented Computing: State of the Art and Research Challenge. IEEE Computer Society. (2007)
[2]Krafzig D, Banke K, Slama D. Enterprise SOA Best Practices. Prentice Hall; 2005.
[3]Schmidt DC. Model-driven Engineering. IEEE Computer, (2006)
[4]Weske M. BPM Concepts, Languages, Architectures. Springer, (2007)
[5]van der Aalst WMP, ter Hofstede A, Weske M. Business Process Management: A Survey. In: International Conference on Business Process Management (BPM0'3), (2003)
[6]Smith H, Fingar P. Business Process Management:The third wave. Meghan-Kieffer, (2003)
[7]Delgado A, Ruiz F, García-Rodríguez de Guzmán I, Piattini M. MINERVA: Model drIveN and sErvice oRiented framework for the continuous business process improVement & relAted tools. In: 5th Int. Workshop on Engineering SO Appls. (WESOA'09), in (ICSOC'09), (2009)
[8]Delgado A, Ruiz F, García-Rodríguez de Guzmán I, Piattini M. Business Process Service Oriented Methodology (BPSOM) with Service generation in SoaML. In: 23rd International Conference on Advanced Information Systems Engineering (CAiSE'11), (2011)
[9]OMG. Query/View/Transformations (QVT), (2008)
[10]Delgado A, Ruiz F, García-Rodríguez de Guzmán I, Piattini M. Model transformations for Business-IT alignment: from collaborative business process to SoaML service model. In: 27th Symposium On Applied Computing (SAC'12), (2012)
[11]OMG. Service Oriented Architecture Modeling Language (SoaML), Beta 2; 2009.
[12]OMG. Business Process Model and Notation (BPMN2), (2011)
[13]Delgado A, Ruiz F, García-Rodríguez de Guzmán I, Piattini M. Towards an ontology for service oriented modeling supporting business processes. In: 4th International Conference on Research Challenges in Information Science (RCIS'10), (2010)
[14]ISO. ISO/IEC 9126 Software engineering – Product quality – Part 1: Quality model; 2001.
[15]Wohlin C, Runeson P, Höst M, Ohlsson M, Regnell B, Wesslén A. Experimentation in Software Engineering: An Introduction. Springer, (2000)
[16]Juristo N, Moreno A. Basics of SE Experimentation. Kluwer Academic Publishers, (2001)
[17]Kitchenham B, Pfleeger S, Pickard L, Jones P, Hoaglin D, El-Emam K, et al. Preliminary guidelines for Empirical Research in SE. Nat. Research Council Canada, Institute for IT, (2001)
[18]Basili VR. Software Modeling and Measurement: The GQM Paradigm. University of Maryland, CS-TR-2956, (1992)
[19]Alarcos. Empirical-WebGen; http://webgen.webportalquality.com/, (2006)
[20] ISO/IEC 25010 Systems and software engineering SQuaRE (Systems and software Quality Requirements and Evaluation) – System and software quality models, (2005-2011)
[21] Delgado, A., Ruiz, F., García-Rodríguez de Guzmán, I., Piattini, M., "Main principles on the integration of SOC and MDD paradigms to BPs: a systematic review", In CCIS series, Springer, In press, (2012). Extended selected article of best papers from ICSOFT'10.
[22]Cruz-Lemus J., Genero M., Manso E., Morasca S., Piattini M., Assessing the understandability of UML statechart diagrams with composite states—A family of empirical studies, Empirical Software Engineering (2009)
[23]Mora B., García F., Ruiz F., Piattini, M. Graphic versus textual software measurement modelling: An empirical study. Software Quality Journal, Vol. 19, Issue 1:201–233, 2011.

Evaluación de la eficiencia de métodos de identificación del defecto de diseño GodClass

Carlos López[1], Esperanza Manso[2], y Yania Crespo[2]

[1] Universidad de Burgos, EPS Edificio C, C/Francisco Vitoria s/n, Burgos, España,
clopezno@ubu.es,
[2] Universidad de Valladolid, Campus Miguel Delibes, Valladolid, España,
manso@infor.uva.es, yania@infor.uva.es

Resumen La identificación de defectos de diseño en entidades de código es una de las tareas del proceso de mantenimiento del software que sirve para evaluar la calidad de un sistema. Un defecto de diseño describe una situación que sugiere un problema potencial en la estructura del software. La intención de diseño de la entidad, que puede ser expresada como estereotipos de clasificadores estándar de UML, proporciona una fuente de información utilizada en algunas definiciones textuales de defectos. En las entidades de código de un sistema software orientado a objetos la información de estereotipos UML no suele estar disponible explícitamente, aunque los diseñadores y programadores la hayan tenido en cuenta en sus soluciones. En la práctica de la automatización de detección de defectos de diseño, esta información es obviada a pesar de su posible utilidad en el proceso de identificación de defectos. Actualmente existen métodos de identificación del defecto de diseño GodClass basados en métricas de código. Incluso existen herramientas que lo automatizan, como InCode y JDeodorant, ambas avaladas con importantes publicaciones de investigación, en las que esta información no se tiene en cuenta. Nosotros proponemos utilizar técnicas de aprendizaje supervisado basado en clasificadores de tipo árbol de decisión, para modelar el problema de la detección de defectos de diseño como una clasificación de entidades de código "con defecto" o "sin defecto". La clasificación inicial en la fase de entrenamiento se puede obtener a partir de los métodos actuales. Este trabajo presenta un caso de estudio para evaluar cómo influye la información relativa a la naturaleza de diseño de la entidad en la detección de defecto GodClass para distintos clasificadores. La evaluación consiste en comparar de medidas de rendimiento del clasificador obtenidas en la fase de entrenamiento (Recuperación, Precisión y F-Measure). Las resultados avalan la validez de considerar la naturaleza de diseño de la entidad en los métodos de identificación de defectos de código.

Palabras clave: defectos de diseño orientados a objetos, experimentación en ingeniería del software, estereotipo de clasificadores UML, aprendizaje supervisado

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 241-254, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

1. Introducción

Un *defecto de diseño* describe una situación que sugiere un problema potencial en la estructura del software. Para decidir si el problema es real o relevante la situación tiene que ser examinada con más detalle en su contexto particular.

En la literatura, este concepto tiene una plétora de términos diferentes para referirse a una familia de conceptos similares: **code smells** o **bad smells** en [1], **disharmonies** en [2], **defects** en [3] o **antipatterns** en [4].

En el área de conocimiento relativa a defectos de diseño, uno de los campos de investigación es la definición de heurísticas o reglas de detección basadas en cierta información de las entidades a evaluar. En esta línea, con este trabajo se busca, por una parte modelar el problema de la detección de defectos mediante clasificadores binarios y, por otra parte, mejorar las reglas de detección incorporando información específica de diseño de la entidad de código que se quiere evaluar.

Los estereotipos de clasificadores estándar de UML (exception,interface, entity, control, test, utility), proporcionan una fuente de información sobre la intención de diseño de la entidad, que puede ser útil en la definición de las heurísticas de detección. Entendemos que el estereotipo UML no es la única información de diseño de la entidad. De ahí que, aunque en este trabajo se utilice el conjunto de etiquetas antes mencionado, podrían incorporarse en el futuro otras etiquetas que expresen de alguna forma la intención de diseño de la entidad. A lo largo del trabajo, para expresar este concepto, se utilizan los términos *naturaleza de diseño de la entidad*, o en su forma abreviada hablaremos de *naturaleza de la entidad*.

Como base de experimentación inicial, para confirmar la influencia de la naturaleza de la entidad en el proceso de detección, hemos elegido el defecto de diseño GodClass. En un sistema software orientado a objetos (OO) una GodClass es un objeto que controla a demasiados objetos en el sistema y ha crecido más allá de toda lógica para convertirse en la clase que lo hace todo. Un caso excepcional de esta descripción, es el participante Mediador en una correcta aplicación del patrón de diseño del mismo nombre. En [5] se indica que un contexto de aplicación del patrón Mediador es el problema del control de dependencias de componentes de una interfaz gráfica. En su apartado de consecuencias se menciona que el participante Mediador es un monolito difícil de mantener y reutilizar, dicho de otro modo es una clase de tipo GodClass. Por tanto, conocer el estereotipo de la entidad, interface en este caso, puede ayudar a eliminar falsos positivos en las heurísticas de detección.

En [2, 3], se definen reglas y heurísticas de detección de este defecto basado en métricas de código. Además existen herramientas, que automatizan esas heurísticas para identificar el defecto GodClass, entre otros. Muchas de estas herramientas están avaladas por importantes publicaciones en el ámbito de la investigación, este es el caso de JDeodorant [6] e InCode [7].

En este trabajo, se propone generar nuevas heurísticas basadas en métricas de código aplicando técnicas de aprendizaje supervisado, basadas en clasificadores de tipo árbol de decisión. La detección de defectos se modela como una clasifi-

cación de entidades de código con defecto o sin defecto. La clasificación inicial necesaria en la fase de entrenamiento se obtiene a partir de las herramientas, JDeodorant e InCode, que identifican los defectos sin considerar la naturaleza de la entidad. Nuestro trabajo se basa en la idea de que "el auditor" encargado de identificar defectos en un sistema, es el conocedor del contexto del problema de la entidad y podría desear incluir información de diseño de la entidad (naturaleza de la entidad) para ser considerada en la tarea de identificación de defectos. Esta nueva funcionalidad puede provocar un cambio en las heurísticas de detección para adaptarse a la nueva información de diseño.

El resto del artículo se estructura de la siguiente manera, en la Sec. 2, se describen los objetivos del estudio. En la Sec. 3 se describe la planificación y en la Sec. 4 se describen cuestiones sobre la operación del estudio: sujetos, objetos y recogida de datos. En la Sec. 5 se presenta un análisis de los datos recogidos y en la última sección se muestran las conclusiones y líneas de trabajo futuras.

2. Definición del caso de estudio

Este estudio se ha realizado siguiendo las recomendaciones dadas en [8, 9].

La pregunta de investigación a responder en este trabajo se puede enunciar como: *¿Influye la naturaleza de la entidad, modelada como estereotipo UML, en la predicción de defectos de diseño tipo GodClass?*.

A partir de dicha pregunta el objetivo principal de este estudio enunciado en formato GQM [10] es el siguiente: *Analizar* entidades de código, *con el propósito de* estudiar cómo impacta el conocimiento de la naturaleza de diseño de la entidad en la detección de defectos de diseño basada en métricas de código, *con respecto* a la eficiencia de la detección, en particular del defecto GodClass, *desde el punto de vista* de los investigadores, *en el contexto* académico de la Universidad de Burgos (UBU) y la Universidad de Valladolid (UVa) y de dos aplicaciones de código abierto JFreeChart 1.0.14 y Eclemma 2.1.0.

Como objetivo secundario del estudio se plantea: *Analizar* entidades de código para comparar dos herramientas de predicción (InCode y JDeodorant) del defecto de diseño GodClass, *con respecto* en la similitud de la clasificación, *desde el punto de vista* de los investigadores, *en el contexto* académico de la Universidad de Burgos (UBU) y la Universidad de Valladolid (UVa) y de dos aplicaciones de código abierto JFreeChart 1.0.14 y Eclemma 2.1.0.

3. Planificación

El tipo de experimento que hemos realizado es un estudio de casos, de acuerdo a la clasificación dada por Robson [11]. En esta sección se presenta el diseño del caso de estudio (Sec. 3.1), proporcionando información relativa de los sujetos y objetos del estudio (Sec. 3.2), y cuestiones relacionadas con la instrumentación para llevar a cabo el estudio (Sec. 3.3).

3.1. Diseño y variables de estudio

Las variables independientes del estudio son, las medidas de la entidad de código y la naturaleza de diseño de la entidad que se describen a continuación:

- Medidas de código orientado a objeto, bien conocidas, de Chidamber y Kemerer [12], Lorenz y Kid [13] y otras. Se trabajará con el siguiente conjunto de métricas: Density of Comments (DC), Executable Statements (EXEC), Total Lines of Code (LOC), Depth in Tree (DIT), Number of Children (NOC), Number of Fields (NOF), Response for Class (RFC), Weighted Methods per Class (WMC), Number of Attributes (NOA), Dependency Inversion Principle (DIP), Lack of Cohesion of Methods (LCOM).
- Naturaleza de la entidad, tiene una escala nominal, cuyas categorías se corresponden con los siguientes estereotipos estandar UML: exception (que se identificará como e_1), interface (e_2), entity (e_3), control (e_4), test (e_5), utility (e_6).

En este tipo de problemas se generan conjuntos de datos no balanceados, también conocidos como desequilibrados. Hay más entidades de código sin defecto que con defecto. En este escenario interesa penalizar la detección de falsos negativos. En minería de datos para problemas similares [14] se propone definir una matriz de costes como posible solución al balanceo de datos. Por ello, hemos considerado otro factor que puede afectar al resultado: el coste de equivocarse en una "no predicción".

- Coste falsos negativos, se mide con escala nominal con las categorías "sin coste", "con coste"

La variable dependiente es:

- La eficiencia de la predicción binaria del defecto GodClass, medida mediante Precisión, Recuperación y F-Measure. La definición de estas medidas se explica más adelante.

Se van a simular dos procesos de aprendizaje supervisado (con naturaleza y sin naturaleza) basado en la generación de clasificadores obtenidos mediante el algoritmo J48 [14]. La clasificación inicial de la entidad en "con defecto" y "sin defecto" se calcula como unión de las dos clasificaciones obtenidas por sendos expertos. En nuestro caso los expertos son JDeodorant e InCode, herramientas que identifican automáticamente el defecto de diseño GodClass.

Los conjuntos de datos resultantes son desequilibrados. Es decir, existen muchas más entidades "sin defecto" que "con defecto". En [15] se recomienda aplicar el algoritmo de clasificación J48 vs C45 para conjuntos de datos desequilibrados con las opciones: no poda, no colapsar y corrección de Laplace.

Medidas de la eficiencia de los clasificadores Las medidas para evaluar la eficiencia de los clasificadores generados, "con naturaleza" y "sin naturaleza" de la entidad, han sido la Recuperación, Precisión y F-Measure descritas en [14].

Para evaluar los posibles resultados cuando se predice una clasificación con dos alternativas (si/no) se utiliza una tabla de contingencia o confusión. En ésta, la diagonal principal recoge las predicciones acertadas (Positivos ciertos TP y Negativos ciertos TN). La otra diagonal recoge los dos posibles errores: Falsos positivos (FP): se produce cuando la salida se predice incorrectamente como "si" (o positivo) cuando realmente es "no" (negativo); Falsos negativos (FN): se produce cuando la salida se predice incorrectamente como "no" (negativo) cuando realmente es"si" (positivo).

		Clase predicha	
		si	no
Clase real	si	Positivos ciertos (TP)	Falsos negativos (FN)
	no	Falsos positivos (FP)	Negativos ciertos (TN)

Las medidas de eficiencia propuestas se definen así:
$Precision = \frac{TP}{TP+FP}$ $Recuperacion = RatiodeTP = \frac{TP}{TP+FN}$

A partir de ellas se puede calcular una única medida conocida como F-measure (media harmónica o media de ratios), y se define como:
$F - measure = \frac{2*Recuperacion*Precision}{Recuperacion+Precision} = \frac{2*TP}{2*TP+FP+FN}$

En el problema de identificación de defectos de código, los dos tipos de error, falsos positivos (FP) y falsos negativos (FN), podrían tener diferentes costes, dependiendo de la organización y el contexto de uso. Es decir, en los escenarios de aplicación del proceso de identificación de defectos sobre un sistema, nos pueden aparecer dos situaciones extremas: se sospecha que muchas de las entidades son defectuosas o, por el contrario, se sospecha que la mayoría no lo son. En el primer caso el coste de un FP, es decir, el coste de identificar como defectuosa una entidad que no lo es, puede penalizarse más que el caso de detectar un FN, clasificar como no defectuosa una que sí que lo es. En el segundo caso, el coste de identificar un FP, puede penalizarse menos que el caso de detectar un FN.

En general en el problema que nosotros estudiamos se presenta la segunda situación, se cuenta con una gran cantidad de entidades donde pocas de ellas son defectuosas. Por ello definimos una matriz de costes para tenerla en cuenta en el proceso de aprendizaje cuando se genera el clasificador. Esto provoca un cambio de reglas de detección en función de los costes. Los subconjuntos de instancias de entrenamiento seleccionados como clase real en la validación cruzada no son aleatorios sino que se crean en función de los costes.

A continuación se muestra la matriz de costes utilizada en este caso de estudio. Se ha considerado ponderar los costes de equivocarse al predecir un falso negativo cinco veces más que el resto. Dicho de otra forma, la organización prefiere mayor recuperación (se decrementa FN) a costa de perder precisión (se incrementa FP). Hemos elegido 5, como peso del coste de los FN por ser el valor, entre los explorados, que proporcionaba una eficiencia mejor (F-Measure).

		Clase predicha	
		si	no
Clase real	si	0 (TP)	5 (FN)
	no	1 (FP)	0 (TN)

Hipótesis Las preguntas que responderemos se derivan de los objetivos de este estudio, propuesto en la Sec. 2. En función de ellas se plantean las correspondientes hipótesis, como se muestra en la Tabla 1.

Tabla 1: Preguntas e hipótesis

n	Pregunta n	$H_{0,n}$
1	¿Influye la naturaleza de la entidad, modelada como estereotipo UML, en la predicción de defectos de diseño tipo GodClass?	$H_{0,1a}$ Utilizar la naturaleza de la entidad (i) no mejora la eficiencia del método de identificación (clasificación) del defecto GodClass en el caso de estudio r. Donde, $r \in$ { JFreeChart 1.0.14, Eclemma 2.1.0. }, $i \in$ { sin naturaleza, con naturaleza } $H_{0,1b}$ Utilizar la naturaleza de la entidad (i) no mejora la eficiencia del método de identificación (clasificación) del defecto GodClass en el caso de estudio r cuando el coste de falsos negativos es k. Donde, $r \in$ { JFreeChart 1.0.14, Eclemma 2.1.0. }, $i \in$ { sin naturaleza, con naturaleza }, $k \in$ { sin coste, con coste FN según la matriz de costes }
2	¿Son similares los métodos respecto a la clasificación binaria del defecto GodClass?	$H_{0,2}$ Los métodos de identificación (j), no se diferencian en cuanto a los resultados de identificación del defecto GodClass en el caso de estudio r. Donde, $r \in$ { JFreeChart 1.0.14, Eclemma 2.1.0. }, $j \in$ { Incode 2.07, JDeodorant 4.0.12 }

Diseño del caso de estudio El diseño propuesto es un diseño cruzado en el que se evalúa la eficiencia de los clasificadores al cruzar las variables naturaleza (n) y coste (c):

- Naturaleza: n con naturaleza y $\overline{n}$ sin naturaleza
- Costes: c con costes y $\overline{c}$ sin costes

	Con costes	Sin costes
Con naturaleza	$MedidasRendimiento_{n,c}$	$MedidasRendimiento_{n,\overline{c}}$
Sin naturaleza	$MedidasRendimiento_{\overline{n},c}$	$MedidasRendimiento_{\overline{n},\overline{c}}$

Una interpretación de los datos de la tabla anterior respecto a la hipótesis principal es: si $MedidasRendimiento_{\overline{n},\overline{c}}$ son iguales o mejores que el resto, entonces no es interesante considerar ni el coste, ni la naturaleza en la identificación de defectos.

3.2. Selección de sujetos y objetos

El sujeto que participa en el estudio es un estudiante de doctorado especializado en el área de mantenimiento del software.

Las bibliotecas Java que son los objetos de este estudio son: JFreeChart 1.0.14, Eclemma 2.1.0. Ambas han sido obtenidas a través del repositorio de software de código abierto *SourceForge* (`http://sourceforge.net/`).

JFreeChart es una biblioteca escrita en Java que facilita a los desarrolladores crear diagramas de calidad en sus aplicaciones. Soporta muchos formatos de salida: componentes Swing, ficheros de imagenes (PNG, JPEG) y ficheros con formato vectorial (PDF, EPS y SVG). Está distribuida bajo licencia GNU-LGPL. Desde el punto de vista de tamaño, la aplicación contiene 976 clases y 244.850 líneas de código.

EclEmma es un plugin de Eclipse escrito en Java para calcular la cobertura de pruebas, y está disponible bajo la licencia Eclipse Public License. Acelera el ciclo de desarrollo de test lanzando los test JUnit a la vez que analiza su cobertura. Mejora el análisis de cobertura, resumiendo los resultados de cobertura y colorea las sentencias del código fuente que no han sido probadas. Desde el punto de vista de tamaño, la aplicación contiene 153 clases y 10.561 líneas de código.

En el estudio, las entidades de código a medir son las clases. Es decir cuando hablamos de número de entidades hablamos de número de clases. La selección de objetos se ha basado en el conocimiento de ambas bibliotecas por el sujeto y en la frecuente utilización por la comunidad de desarrolladores en Java.

3.3. Instrumentación

En el estudio hemos utilizado las herramientas siguientes:

- Una herramienta para calcular métricas de las entidades de código, RefactorIt. En el contexto de predicción de defectos de diseño basados en métricas

que se plantea en este trabajo se necesita una herramienta que calcule un amplio conjunto de métricas orientadas a objetos y permita la exportación de resultados para su análisis posterior. Bajo estas premisas, la herramienta seleccionada para obtener las medidas ha sido RefactorIt. En la Sec 3.1 se describió el conjunto de las métricas a emplear en el estudio, todas ellas proporcionadas por la herramienta.

- Una herramienta desarrollada por los propios autores [16] que ayuda a la clasificación de entidades de código según su intención de diseño, expresada como estereotipo estándar de UML (exception, interface, entity, control, test, utility)
- Dos plugins de Eclipse, Incode 2.07 [7] y JDeodorant 4.0.12 [6], que permiten obtener una predicción del defecto GodClass sobre las entidades de código de los objetos seleccionados en este caso de estudio. Se han seleccionado, Incode 2.07 y JDeodorant 4.0.12, por la buena documentación de sus métodos reflejada en sus artículos de investigación publicados.
- Una herramienta de minería de datos, Weka 3.7.5 que permite generar clasificadores con diferentes métodos de caja blanca (JRIP, J48) y calcula las medidas del proceso de aprendizaje supervisado (Recuperación, Precisión, F-Measure). La herramienta se ha elegido, por cumplir estos requisitos y por su documentación [14].
- Una herramienta estadística, R 2.7.1, para realizar el test de hipótesis de McNemar.

4. Operación

En este apartado se presenta cómo se ha llevado a cabo el estudio.

La instrumentación ha sido realizada en un único ordenador personal en el mes de febrero de 2012. La descripción del entorno tecnológico es la siguiente: Intel(R) Core(TM) 2Quad CPU Q8300 2.5 GHz, con RAM 4GB, Sistema Operativo Windows 7 Profesional, Eclipse 3.7.1, Java 1.7.0.

La principal incidencia encontrada ha sido un problema de eficiencia de memoria cuando se trabaja con el plugin de Eclipse JDeodorant, especialmente relevante en el caso de analizar proyectos grandes. Para superar el problema se optó por ampliar la memoria de ejecución de la máquina virtual de Java que lanzaba Eclipse (-Xmx1024m) o realizar la recogida por partes de cada módulo separado de la aplicación.

5. Análisis

Las dos subsecciones siguientes, presentan el análisis para cada una de las hipótesis de la Sec. 3 obtenidas de los objetivos de investigación expuestos en la Sec. 2.

5.1. Estudio de la eficiencia de los clasificadores para identificar GodClass

En este apartado se estudia la hipótesis $H_{0,1}$ analizando de forma descriptiva las medidas de eficiencia de los clasificadores generados en el proceso de aprendizaje, la Recuperación, la Precisión y F-Measure.

$H_{0,1a}$ Utilizar la naturaleza de la entidad (i) no mejora la eficiencia del método de identificación (clasificación) del defecto GodClass en el caso de estudio r.
Donde,
$r \in \{$ JFreeChart 1.0.14, Eclemma 2.1.0. $\}$,
$i \in \{$ sin naturaleza, con naturaleza $\}$

La estructura del análisis para la hipótesis $H_{0,1a}$ es la siguiente:

1. Analizar cómo influye en la eficiencia de la identificación del defecto GodClass utilizar o no la naturaleza de la entidad, en el caso de estudio JFreeChart 1.0.14.
2. Analizar cómo influye en la eficiencia de la identificación del defecto GodClass utilizar o no la naturaleza de la entidad, en el caso de estudio Eclemma 2.1.0.
3. Comparar los resultados anteriores.

Todos los resultados obtenidos se pueden consultar en: `http://dl.dropbox.com/u/18996787/JISBD2012/4DCAnalisisEficienciaCompleto.pdf`.

1. En el caso de estudio JFreeChart, las medidas de eficiencia, con respecto a la clase True (entidades con el defecto GodClass), mejoran levemente cuando consideramos la naturaleza de la entidad, mientras que la recuperación se mantiene igual. Los datos que justifican la afirmación son los siguientes:
   Sin naturaleza: Precisión (0,581) Recuperación (0,537) F-Measure (0,558)
   Con naturaleza: Precisión (0,600) Recuperación (0,537) F-Measure (0,567)
2. En el caso de estudio Eclemma, las medidas de eficiencia, con respecto a la clase True (entidades con el defecto GodClass), mejoran significativamente cuando consideramos la naturaleza de la entidad.
   Sin naturaleza: Precisión (0,200) Recuperación (0,182) F-Measure (0,190)
   Con naturaleza: Precisión (0,455) Recuperación (0,455) F-Measure (0,455)
3. En conclusión, utilizar la naturaleza de la entidad ha mejorado las medidas de eficiencia de detección de defectos GodClass.

$H_{0,1b}$ Utilizar la naturaleza de la entidad (i) no mejora la eficiencia del método de identificación (clasificación) del defecto GodClass en el caso de estudio r cuando el coste de falsos negativos es k.
Donde,
$r \in \{$ JFreeChart 1.0.14, Eclemma 2.1.0. $\}$,
$i \in \{$ sin naturaleza, con naturaleza $\}$,
$k \in \{$ sin coste, con coste FN según la matriz de costes $\}$

La estructura del análisis para la hipótesis $H_{0,1b}$ es la siguiente:

1. Analizar cómo influye en la eficiencia de la identificación del defecto GodClass utilizar o no la matriz de costes, al utilizar o no la naturaleza de la entidad, en el caso de estudio JFreeChart 1.0.14.
2. Analizar cómo influye en la eficiencia de la identificación del defecto GodClass utilizar o no la matriz de costes, al utilizar o no la naturaleza de la entidad, en el caso de estudio Eclemma 2.1.0.
3. Comparar los resultados anteriores.

1. En el caso de estudio JFreeChart, las medidas de eficiencia cuando utilizamos la matriz de costes, con respecto a la clase True (entidades con el defecto GodClass), mejoran levemente al considerar la naturaleza de la entidad.
   Sin naturaleza: Precisión (0,570) Recuperación (0,970) F-Measure (0,718)
   Con naturaleza: Precisión (0,575) Recuperación (0,970) F-Measure (0,722)
   Las medidas de eficiencia cuando utilizamos la naturaleza de la entidad, con respecto a la clase True (entidades con el defecto GodClass), mejoran al considerar la matriz de costes para la recuperación y F-Measure, cuyo aumento compensa la disminución de la precisión.
   Sin matriz de costes: Precisión (0,600) Recuperación (0,537) F-Measure (0,567)
   Con matriz de costes: Precisión (0,575) Recuperación (0,970) F-Measure (0,722)
   Las medidas de eficiencia cuando no utilizamos la naturaleza de la entidad, con respecto a la clase True (entidades con el defecto GodClass), mejoran al considerar la matriz de costes para la recuperación y F-Measure cuyo aumento compensa la disminución de la precisión.
   Sin matriz de costes: Precisión (0,581) Recuperación (0,537) F-Measure (0,558)
   Con matriz de costes: Precisión (0,570) Recuperación (0,970) F-Measure (0,718)
2. En el caso de estudio Eclemma, las medidas de eficiencia cuando utilizamos la matriz de costes, con respecto a la clase True (entidades con el defecto GodClass), mejoran significativamente cuando consideramos la naturaleza de la entidad.
   Sin naturaleza: Precisión (0,250) Recuperación (0,364) F-Measure (0,296)
   Con naturaleza: Precisión (0,385) Recuperación (0,455) F-Measure (0,417)
   Las medidas de eficiencia cuando utilizamos la naturaleza de la entidad, con respecto a la clase True (entidades con el defecto GodClass), no mejoran, la F-Measure empeora ligeramente.
   Sin matriz de costes: Precisión (0,455) Recuperación (0,455) F-Measure (0,455)
   Con matriz de costes: Precisión (0,385) Recuperación (0,455) F-Measure (0,417)
   Las medidas de eficiencia cuando no utilizamos la naturaleza de la entidad, con respecto a la clase True (entidades con el defecto GodClass), mejoran al considerar la matriz de costes.
   Sin matriz de costes: Precisión (0,200) Recuperación (0,182) F-Measure (0,190)
   Con matriz de costes: Precisión (0,250) Recuperación (0,364) F-Measure (0,296)
3. Teniendo en cuenta que la cantidad de entidades con defectos es muy pequeña, se pone claramente de relieve que la matriz de costes es necesaria: es significativo que en presencia de la matriz de costes, utilizar también la naturaleza de la entidad ha mejorado las medidas de la eficiencia para detectar el defecto GodClass en ambos casos de estudio. La conclusión a la vista de todos los

resultados (Tabla 2), es que tanto la matriz de costes como la naturaleza de la entidad son dos factores relevantes para mejorar la detección de dicho defecto.

La Tabla 2 resume los resultados del análisis: Con o Sin Naturaleza (n, $\overline{n}$), Con o Sin Costes (c, $\overline{c}$).

	Precisión	Recuperación	F-Measure
$\overline{c}$: n vs. $\overline{n}$	Mejora en ambos casos	Mejora en ambos casos	Mejora en ambos casos
$\overline{n}$: c vs $\overline{c}$	Mejoran en ambos casos	Mejora en ambos casos	Mejora en ambos casos
n: c vs $\overline{c}$	JFreeChart mejora	JFreeChart mejora	JFreeChart mejora
c: n vs. $\overline{n}$	Mejora en ambos casos	Mejora en ambos casos	Mejora en ambos casos

Tabla 2: Resumen del análisis respecto $H_{0,1a}$ y $H_{0,1b}$

5.2. Comparación de los métodos de clasificación

En este apartado se estudia la hipótesis $H_{0,2}$ utilizando el test estadístico de McNemar.

La prueba no paramétrica de McNemar sirve para comparar las puntuaciones de dos jueces en si/no sobre k objetos. [17]. Donde,

- Los objetos a clasificar son las entidades de código de la aplicación
- La variable dependiente es si la entidad tiene o no tiene el defecto GodClass

$H_{0,2}$ Los métodos de identificación (j), no se diferencian en cuanto a los resultados de identificación de GodClass en el caso de estudio r.
Donde,
$r \in$ { JFreeChart 1.0.14, Eclemma 2.1.0. },
$j \in$ { Incode 2.07, JDeodorant 4.0.12 }

La Tabla 3 contiene la información sobre la identificación de defectos de los casos de estudio con los dos métodos utilizados.

En la siguiente tabla se muestra el resumen estadístico utilizado para contrastar la hipótesis nula $H_{0,2}$. A partir del test de hipótesis de McNemar se rechaza la hipótesis nula ($\alpha <= 0{,}05$) en las dos aplicaciones consideradas. Así que podemos concluir que los métodos de Incode y JDeodorant no son similares. Estos resultados justifican la necesidad de adaptar el método de identificación por medio de aprendizaje supervisado.

	Nº total de entidades	Nºentidades con defectos identificadas InCode	Nºentidades con defectos identificadas JDeodorant	Nºentidades con defectos identificadas JDeodorant e InCode
JFreechart	975	67	0	67
eclemma	152	0	6	6

Tabla 3: Predicciones del defecto GodClass con JDeodorant e Incode

	JFreeChart	Eclemma
p-valor	7.433e-16	0.04123
McNemar	65.0149	4.1667

5.3. Amenazas a la validez

La validez de construcción del caso de estudio, entendida como las variables que han sido correctamente medidas, está amenazada por la variable naturaleza de la entidad, pues la clasificación de entidades en estereotipos UML no es ortogonal cuando se aplica sobre entidades de código reales.

La validez interna del caso de estudio, entendida como la causalidad de nuestras conclusiones, está afectada porque la clasificación de entidades en estereotipos UML es subjetiva.

Desde la perspectiva de la validez externa, referida a cuántos de nuestros resultados se pueden generalizar en otras circunstancias, se han identificado las siguientes amenazas:

- Los proyectos utilizados y el lenguaje de programación de los mismos limita la extensión de estos resultados.
- La selección de los sujetos se ha hecho por conveniencia.
- Las medidas de eficiencia de los clasificadores parecen depender del algoritmo utilizado y su configuración para clasificar, en nuestro caso J48 -O -U -A.
- La variación en la matriz de costes puede modificar los resultados, en nuestro caso se ha elegido un peso de 5.

6. Conclusiones

El objetivo principal de este estudio es comprobar si la naturaleza de la entidad de código, entendida como estereotipos estandar de UML, mejora la eficiencia de los métodos de identificación de GodClass, basados en métricas de código. Este objetivo es de interés porque las técnicas de identificación de defectos en entidades de código son útiles para mejorar el mantenimiento del mismo.

Durante el estudio se ha observado que en el problema de identificación de defectos sobre entidades de código, se generan conjuntos de datos no balanceados.

Para resolver este problema, hemos considerado el coste del error de los falsos negativos en los métodos de detección.

Podemos concluir lo siguiente:

1. En el estudio de la eficiencia de los clasificadores para identificar el defecto GodClass, los resultados observados indican que la naturaleza de diseño de la entidad mejora su eficiencia. Además, la matriz de costes, cuando se penaliza el coste de los falsos negativos, también mejora dichas medidas. Por ello, la matriz de costes y la naturaleza de diseño de la entidad son dos factores relevantes para mejorar dicha eficiencia.
2. En el estudio de comparación de los dos métodos de identificación del defecto GodClass, los resultados indican que no existe concordancia, por tanto la clasificación tiene un grado de subjetividad. Es por ello por lo que se aboga desde este trabajo porque los métodos de identicación de GodClass sean adaptados al contexto de la aplicación con técnicas de aprendizaje supervisado.

Como líneas de trabajo futuro, para completar los resultados de este estudio estamos trabajando en:

- Añadir la supervisión humana para validar los resultados de los métodos eliminando los falsos positivos en función de la excepciones documentadas en [1].
- Repetir el caso de estudio sobre otros defectos de diseño como DataClass y FeatureEnvy, por ejemplo.
- Estudiar posibles refinamientos de la clasificación de entidades de código en esterotipos UML.

Referencias

1. Martin Fowler, Kent Beck, John Brant, William Opdyke, and Don Roberts. *Refactoring: Improving the Design of Existing Code*. Object Technology Series. Addison-Wesley, June 1999.
2. Michele Lanza and Radu Marinescu. *Object-Oriented Metrics in Practice - Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems*. Springer, 2006.
3. Naouel Moha, Yann-Gaël Guéhéneuc, Anne-Françoise Le Meur, Laurence Duchien, and Alban Tiberghien. From a domain analysis to the specification and detection of code and design smells. *Formal Aspects of Computing*, May 2009.
4. William J. Brown, Raphael C. Malveau, Hays W. McCormick III, and Thomas J. Mowbray. *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis*. John Wiley and Sons, March 1998.
5. Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns*. Addison-Wesley Professional, January 1995.
6. Marios Fokaefs, Nikolaos Tsantalis, Eleni Stroulia, and Alexander Chatzigeorgiou. Jdeodorant: identification and application of extract class refactorings. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 1037–1039, New York, NY, USA, 2011. ACM.

7. Radu Marinescu and George Ganea. incode.rules: An agile approach for defining and checking architectural constraints. In *Proceedings of the Proceedings of the 2010 IEEE 6th International Conference on Intelligent Computer Communication and Processing*, ICCP '10, pages 305–312, Washington, DC, USA, 2010. IEEE Computer Society.
8. C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering: An Introduction*, volume 6 of *International Series in Software Engineering*. Springer, 2000.
9. Per Runeson and Martin Host. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14:131–164, 2009.
10. Victor Basili, Gianluigi Caldiera, and Dieter H. Rombach. The goal question metric approach. In J. Marciniak, editor, *Encyclopedia of Software Engineering*. Wiley, 1994.
11. C. Robson. *Real World Research - A Resource for Social Scientists and Practitioner-Researchers*. Blackwell Publishing, Malden, second edition, 2002.
12. Shyam R. Chidamber and Chris F. Kemerer. Towards a metrics suite for object oriented design. *ACM SIGPLAN Notices*, 26(11):197–211, 1991.
13. Mark Lorenz and Jeff Kidd. *Object-Oriented software metrics: a practical guide*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1994.
14. Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Elsevier, January 2011.
15. David Cieslak, T. Hoens, Nitesh Chawla, and W. Kegelmeyer. Hellinger distance decision trees are robust and skew-insensitive. *Data Mining and Knowledge Discovery*, 24(1):136–158, 2012. 10.1007/s10618-011-0222-1.
16. Carlos López, Esperanza Manso, and Yania Crespo. The identification of anomalous code measures with conditioned interval metrics. In *13th TOOLS Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE 2010) Málaga, Spain*, Málaga, July 2010.
17. S. Siegel and N.J. Castellan. *Non-parametric statistics for the behavioural sciences (2nd ed)*. Mc Graw Hill, 1988.

Alf como lenguaje de especificación de refactorizaciones

Raúl Marticorena[1] y Yania Crespo[2]

[1] Universidad de Burgos, EPS Edificio C, C/Francisco Vitoria s/n, Burgos, España,
rmartico@ubu.es,
[2] Universidad de Valladolid, Campus Miguel Delibes, Valladolid, España,
yania@infor.uva.es

Resumen Las definiciones más habituales de operaciones de refactorización se centran en un lenguaje particular. Sin embargo, una de las líneas de investigación en refactorización en los últimos años aborda la independencia del lenguaje en la definición de refactorizaciones, que posteriormente deberá ser implementada para lenguajes particulares. Se han aportado diferentes soluciones para representar el código con cierta independencia del lenguaje, sin embargo la definición e implementación de refactorizaciones continúa siendo mayoritariamente dependiente del lenguaje. En cuanto a esta definición e implementación de refactorizaciones, se aprecia un importante salto bien sea desde las definiciones de refactorizaciones en forma de recetas, bien sea desde las definiciones formales utilizando lógica de predicados o reescritura de grafos, a la implementación real de la refactorización.
En este trabajo, tomando como base un metamodelo UML definido para la representación de código de forma independiente del lenguaje, se valida la aplicación del lenguaje de acciones Alf como lenguaje para la definición de refactorizaciones. El uso de Alf tiene una doble implicación. Por una parte se consigue acercar la definición de refactorizaciones a su posterior implementación, dado que se produce una traducción con menor esfuerzo desde el lenguaje de acciones al lenguaje de implementación elegido. Por otra parte, el uso de Alf y los parsers asociados, permiten validar que el metamodelo contiene la información suficiente para definir refactorizaciones.

Palabras clave: refactorización, lenguaje de modelado UML, metamodelo, lenguaje de acciones Alf

1. Introducción

Una refactorización se puede definir como una transformación del software que facilita la comprensión de su diseño sin alterar su comportamiento observable [1]. Al abordar el problema de su definición, construcción y aplicación es necesario acotar el nivel de abstracción al que se va a trabajar. En general, aunque no de manera exclusiva, las refactorizaciones se centran a nivel de implementación sobre lenguajes orientados a objetos, transformando el código fuente escrito en

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 255-268, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

un lenguaje de programación concreto o *lenguaje objetivo*. Una de las líneas de investigación abierta en los últimos años aborda la independencia del lenguaje en la definición de refactorizaciones, que posteriormente deberá ser aplicada a diferentes lenguajes objetivo.

Definir, implementar y aplicar una refactorización de manera completamente independiente al lenguaje es imposible, como se estableció en [2,3]. Esto se debe a que la refactorización es aplicada en última instancia sobre código real, con sus características particulares. Éstas influyen tanto en el cumplimiento de las precondiciones como en las transformaciones o acciones necesarias.

Sin embargo, a pesar de las diferencias, se pueden encontrar algunos puntos en común. Una vez acotado el conjunto de lenguajes objetivo, existen conceptos básicos comunes a todos. En el caso de lenguajes de programación orientados a objetos (LPOO) estáticamente tipados se tienen espacios de nombres, tipos, clases, atributos, métodos, herencia, genericidad, etc.

En base a esta idea se pueden definir las refactorizaciones de forma general, empleando conceptos más abstractos, para posteriormente particularizar su solución al lenguaje objetivo. Este tipo de cuestiones también se han afrontado con éxito en la implementación de los patrones de diseño en diferentes lenguajes. El patrón se esboza de forma general, pero la forma de llevarse a cabo puede ser diferente en cada lenguaje (*idioms*).

El principal problema a la hora de llevar a cabo la implementación final de la refactorización es la existencia de un cierto desajuste o distancia entre la definición, ya sea formal, semiformal o texto libre, y la implementación concreta de la refactorización para el lenguaje objetivo, dependiendo a su vez del lenguaje con el que se desarrolla.

Con el objetivo de dar una respuesta este problema, en la Sec. 2, se expone el problema de la independencia de lenguaje en la definición y construcción de refactorizaciones tal y como se ha abordado en otros trabajos. Posteriormente, en la Sec. 3 se establece la base de nuestro trabajo, describiendo el metamodelo básico propuesto. A partir de dicha propuesta, se razona en la Sec. 4, la adecuación de Alf como lenguaje de especificación de refactorizaciones, cubriendo las deficiencias detectadas en otras soluciones. En la Sec. 5 se presenta el caso de estudio de la aplicación de Alf sobre refactorizaciones bien conocidas, validando la propuesta de este trabajo, y determinando los pros y contras frente a otras soluciones. Finalmente, en la Sec. 6 se presentan las conclusiones y líneas de trabajo futuro.

2. Trabajos relacionados

La independencia del lenguaje ha sido tratada en diferentes trabajos, siempre con el objetivo de abordar un gran número de refactorizaciones común al mayor conjunto de lenguajes posible. En [4], se introdujo la noción de refactorización genérica de programas, como propuesta inicial hacia un *framework* parametrizable para cada lenguaje. Se sugirió su implementación con Haskell, proporcionando puntos de enganche para los elementos específicos para cada lenguaje. El

framework podía ser instanciado para diferentes lenguajes. Como el propio autor apuntó, se trataba de un esquema inicial apuntando los conceptos fundamentales que se deben incluir y parametrizar, tomando como caso de estudio la extracción de abstracciones con el *framework*. En la medida de nuestro conocimiento no se ha llevado a cabo ni se dispone de una implementación concreta.

Otras aproximaciones, menos ambiciosas a la hora de tratar el problema de la independencia del lenguaje están basadas en metamodelos. En [5,6], se definió el metamodelo FAMIX para almacenar información en varias herramientas CASE para reingeniería. Este metamodelo especifica las entidades principales en orientación a objeto: clases, métodos, atributos, herencia, etc. Para poder completar el proceso de reingeniería, introducen dos entidades adicionales: la invocación y el acceso a propiedades. En FAMIX no hay tratamiento para el concepto de herencia múltiple ni genericidad.

Siguiendo el metamodelo de FAMIX, en [7, 8] se expone el entorno de reingeniería denominado Moose. El entorno se basa en la versión 2.0 de FAMIX. En la actualidad están trabajando sobre una versión 3.0 ante la diversidad de variaciones que se han producido sobre dicho metamodelo. El entorno Moose trabaja con instancias del metamodelo de FAMIX recogidas del código fuente. Hay que señalar que la información extraída es el subconjunto mínimo necesario para poder realizar los análisis y validar las precondiciones de las refactorizaciones. La información obtenida del cuerpo de los métodos es parcial, por lo que las refactorizaciones dependientes de dicha información no pueden llevarse a cabo.

En [9] se describió el motor de refactorizaciones utilizado, indicando dos puntos claves. En primer lugar, el análisis de las precondiciones se lleva a cabo sobre las instancias de FAMIX. En segundo lugar, las modificaciones sobre el código se realizan, dependiendo del lenguaje analizado, con herramientas externas. El análisis de las precondiciones y la ejecución de las transformaciones se realiza de forma diferente, complicando la arquitectura y perdiendo potencial de reutilización.

En la misma línea que FAMIX, una solución basada en metamodelos se propuso en [2, 10]. A través de la extensión del metamodelo de UML 1.4 con ocho modificaciones adicionales independientes del lenguaje, se definió un nuevo metamodelo denominado GrammyUML, con el objetivo de resolver las inconsistencias entre las refactorizaciones realizadas sobre modelos de diseño y código. Se definieron las refactorizaciones con OCL, validando la solución sobre dos refactorizaciones (**Pull Up Method** y **Extract Method**), de forma teórica. La propuesta no ha sido renovada en posteriores trabajos, y en la medida de nuestro conocimiento, no existen implementaciones actuales.

En otros trabajos como [3] se busca un alto grado de independencia del esquema y tecnología empleada a través de XML. Se utiliza XML como mecanismo de transformación. Sin embargo, pese a lo atractivo de utilizar un metalenguaje de uso general como XML, las consultas y actualizaciones que conforman una refactorización deben ser reescritas para cada lenguaje objetivo, limitando las posibilidades de reutilización.

Otra línea de trabajos relacionada, aborda la definición de refactorizaciones mediante transformaciones de grafos [11]. Los programas se expresan como *grafos*, las refactorizaciones se corresponden con *reglas de producción del grafo*, y las pre/postcondiciones de la refactorización, como *pre y poscondiciones de aplicación.* Aunque utilizando otro formalismo, los elementos generales se mantienen, pero vuelve a existir un salto hacia la solución final de implementación.

Concluyendo, las soluciones basadas en metamodelos pueden dar una respuesta a la definición y aplicación de refactorizaciones con cierto grado de independencia del lenguaje, pero todavía es necesario dar una solución de continuidad entre el análisis realizado en la refactorización, la definición de precondiciones y acciones, y la implementación final.

3. Independencia del lenguaje con el metamodelo MOON

En [12] se presentó la gramática de una notación minimal para lenguajes orientados a objetos denominada MOON. Con MOON se definió un marco de referencia específico para desarrollar refactorizaciones con el objetivo de transformar elementos de implementación, escritos en un lenguaje de programación orientado a objetos, estáticamente tipado y con genericidad.

En MOON se representan: declaración de tipos, clases, creación de objetos, envío de mensajes, constantes manifiestas, asociaciones de objetos a entidades (instrucciones de creación, asignación, y paso de expresiones en envío de mensajes), herencia simple/múltiple y genericidad (con variantes del sistema de tipos de subtipado –incluyendo acotación F–, cláusulas *tal que* (*where clauses*) y conformidad).

Partiendo de la gramática, se trabajó en el diseño de un metamodelo MOON que recoge todos los elementos y relaciones definidos. A partir de instancias del metamodelo se obtiene una representación objetual del código escrito en el lenguaje objetivo a refactorizar.

A continuación, se presenta el modelado de las características fundamentales, a través de abstracciones, definidas en el lenguaje MOON. Las características variables o particulares son soportadas para cada lenguaje objetivo como puntos de extensión.

3.1. Jerarquía principal de MOON

La jerarquía principal del metamodelo MOON (ver Fig. 1) se deduce no sólo de la gramática de MOON, sino también de la semántica de los objetos MOON, con el objetivo claro de construir un *framework* extensible basado en el metamodelo.

En nuestro metamodelo la clase raíz única del metamodelo es `ObjectMoon`. Todos los descendientes de esta clase contienen un identificador único (`id`). La evolución del metamodelo se daría a través de los mecanismos de herencia, en el caso de tener que incorporar nuevos conceptos siempre y cuando se justifique la generalidad del nuevo concepto introducido, manteniendo el objetivo fundamental de mantener una representación minimal para una familia amplia de lenguajes.

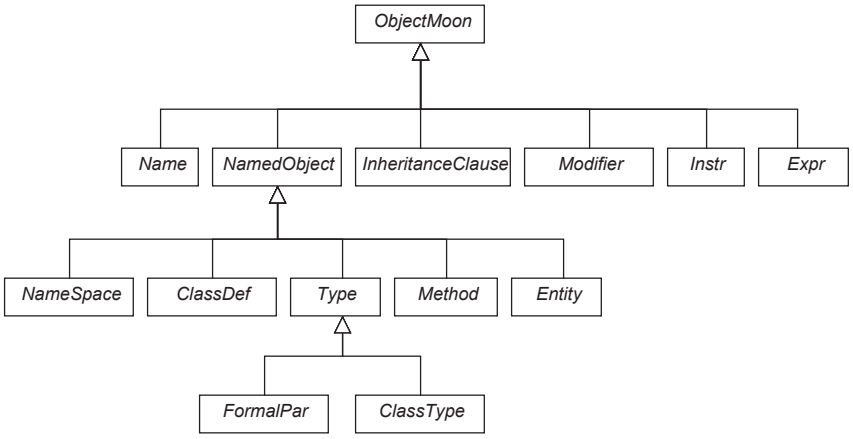


Figura 1: Jerarquía principal en MOON

3.2. Información disponible en MOON

En las refactorizaciones es necesario conocer las dependencias existentes entre los elementos representados, para poder determinar su viabilidad. Estas dependencias vienen dadas por relaciones de tipos, genericidad, herencia, asociaciones y relaciones de dependencia, derivadas a partir de las clases en la Fig.1.

- Las relaciones entre tipos y clases, y su organización lógica en contextos o espacios de nombres, se derivan a partir de las clases `NameSpace`, `ClassDef` y `Type` como elementos fundamentales.
- Las relaciones de genericidad se reflejan en la introducción de parámetros de tipo `FormalPar` y características adicionales sobre clases y tipos generados por las clases `ClassDef` y `ClassType` respectivamente.
- La relaciones de herencia entre clases y tipos son representadas a través de las clases `InheritanceClause`, así como sus modificadores de herencia a través de la clase `Modifier`.
- Las entidades, con sus tipos asociados y las signaturas de métodos, determinan las operaciones a realizar a través de otras entidades, y se corresponden con las clases `Entity` y `Method`.
- Las instrucciones `Instr` y las expresiones `Expr` dan lugar a relaciones de dependencia, a través del cuerpo de los métodos, poniendo en combinación todos los elementos previos.

Dentro de esta jerarquía, se distinguen un conjunto de elementos con nombre (`NamedObject`) que se caracterizan por poseer un nombre simple (`Name`) y un nombre único calculado a partir del nombre simple y del nombre del contexto en el que se encuentran. El nombre simple sirve para identificar al elemento dentro

de un contexto y el nombre único sirve para identificar al elemento, de forma única en el sistema.

Se ha llegado a un metamodelo final que consta de 50 clases e interfaces, manteniéndose en un número medio aceptable. El metamodelo recoge los elementos claves para recoger la semántica de una familia extensa de LPOO estáticamente tipados, con un número manejable de clases que permiten su extensión a lenguajes concretos de la corriente principal pudiendo abstraer cuestiones relativas a la refactorización del código en muchas facetas.

4. Características de Alf para la definición de refactorizaciones

En los catálogos clásicos de refactorizaciones como [13, 14], se razonaba a partir de la información extraída del código a través de una lógica de predicados. Esta solución es adaptable a una solución basada en metamodelos. Puede tomarse como base el trabajo de [15] donde se planteaba la transformación de diagramas de clases (incluyendo también expresiones OCL) a una lógica de predicados de primer orden con tipos. Con estos trabajos se puede asegurar la corrección de las precondiciones y postcondiciones, pero no la forma de definir las acciones de transformación.

Partiendo de estas experiencias, establecida la base sólida de MOON como metamodelo, en este trabajo se estudia Alf como formalismo para la definición de las refactorizaciones. El formalismo utilizado debe validar la adecuación del metamodelo MOON para permitir las consultas necesarias sobre la información disponible del código, así como las acciones que modifican el estado de los objetos del metamodelo.

El lenguaje Alf es el *Action Language for Foundational UML* [16]. Es una representación textual de elementos de modelado UML. La semántica de ejecución se establece por una correspondencia de su sintaxis concreta a la sintaxis abstracta del estándar *Foundational* UML (fUML) [17]. Así pues, el resultado de la ejecución de un fichero de entrada Alf está dado por la semántica del modelo fUML, definido en su especificación.

Frente a las limitaciones que impone un lenguaje de consulta como OCL, el lenguaje Alf proporciona un conjunto de características mucho más amplio. Sin dar una descripción exhaustiva de Alf decimos que el lenguaje se adecúa a la definición de refactorizaciones sobre un metamodelo descrito en UML porque:

- Permite la representación estructural completa de un modelo (*e.g.* clases, asociaciones, multiplicidades, restricciones, etc.)
- Permite la navegación sobre los objetos del modelo, sus asociaciones y el acceso/manipulación de las instancias.
- Permite la especificación de comportamientos ejecutables en dichos modelos representados con UML (*e.g.* actividades como elemento fundamental en Alf [16], etc.)

- Establece un sistema implícito de tipos (no requiere declaración explícita de tipos en una actividad) pero siempre proporciona comprobación de tipos en las operaciones realizadas (*e.g.* asignaciones, paso de mensaje, etc.)
- Incluye expresiones, y más concretamente expresiones de expansión sobre secuencias de elementos permitiendo cuantificadores (*e.g.* `forAll`, `exists`) y consultas (*e.g.* `select`). Esto permite definir consultas de forma declarativa sobre el modelo con equivalencia a una lógica de predicados de primer orden.
- Incluye el concepto de sentencia y bloque de sentencias, junto un conjunto amplio de sentencias predefinidas equivalente a instrucciones de control de flujo (*e.g.* `if`, `for`, `switch`), permitiendo también una definición imperativa de actividades.
- Define un conjunto básico de tipos (*e.g.* `Boolean`, `String`, `Integer`, etc.) y bibliotecas predefinidas de estructuras de datos genéricas (*e.g.* `Collection`, `List`, `Set`, etc.)

En la Fig. 2 se muestra el proceso completo del uso de Alf propuesto para la definición y construcción de refactorizaciones. Partiendo del metamodelo MOON, se define en ficheros de texto con sintaxis Alf (extensión `.alf` por convención) tanto el metamodelo, como las actividades que realizan consultas (funciones y predicados) y que modifican el estado de los objetos (acciones). La validación se realiza a través de un parser implementado en Java [18] para chequear la corrección y la no violación de restricciones del lenguaje.

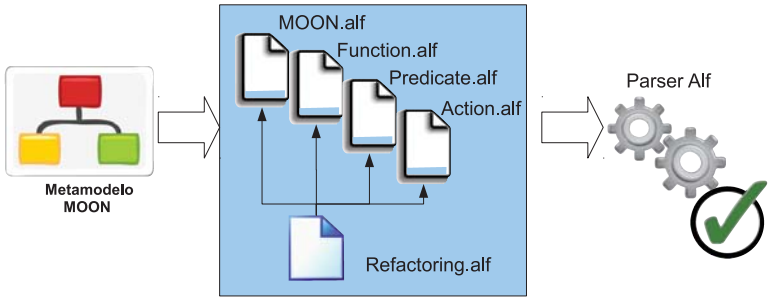


Figura 2: Flujo de trabajo usando Alf en la definición de refactorizaciones

En la siguiente sección se muestra el uso de Alf como lenguaje de representación para el conjunto de funciones, predicados y acciones a partir de los cuales se puede definir un catálogo de refactorizaciones. Esto permite analizar las ventajas y desventajas encontradas en la utilización de dicho lenguaje.

5. Casos de estudio utilizando Alf

En esta sección se presentan dos casos de estudio sobre refactorizaciones bien conocidas, utilizando como lenguaje para su especificación el lenguaje de acciones

ALF. Abarcando cambios que afectan a un conjunto de clases cliente (**Rename Class**) o a clases clientes y descendientes en una jerarquía de herencia (**Push Down Field**).

5.1. Refatorización Rename Class

Esta refactorización fue descrita en [13, 14], y aunque no está definida explícitamente en el catálogo de [1], ha sido incluida en la práctica totalidad de entornos de desarrollo dada su utilidad.

En la siguiente tabla se muestra la definición de la refactorización **change_class_name** que se encuentra en [13, Capítulo 5, Pág. 42]:

Nombre	**Change_class_name**
Descripción	Cambiar el nombre de una clase.
Argumentos	clase (C) y string (S)
Precondiciones	- El nuevo nombre no colisiona con una clase ya existente: $\forall class \in Program.classes, class.name \neq S$
Preservación del comportamiento	La precondición asegura nombres distintos de clases (propiedad dos [13]). Cambiar el nombre de una clase no cambia su comportamiento (satisface propiedad siete). El resto de propiedades se preservan.
Efectos	El cambio de nombre es reflejado a lo largo del programa (*i.e.* en clases y declaraciones de subclases, constructores y destructores, y en la instanciación de la clase).

En [14, Capítulo 3, Pág. 27] la definición es menos detallada.

RenameClass(class, newName) Renombra una clase y actualiza todas las referencias. No puede existir ninguna clase o nombre global *newName* para que pueda ser legal la refatorización.

La refactorización no se detalla más, quedando a expensas de la utilización de un catálogo de lo que autor denomina funciones de análisis primitivas y derivadas para consultar y funciones de transformación para modificar el código. Sin embargo el catálogo de funciones dado, no parece completo, quedando pendiente su compleción para poder definir la refactorización de manera más precisa.

La definición de refactorizaciones con lógica de predicados es una de las soluciones tradicionales, dotando a éstas de una mayor base formal que las definiciones en base a "recetas" o colección de mecanismos [1]. En trabajos previos se ha utilizado dicha técnica [19, 20] para completar una plantilla de definición de refactorización, a partir de la información del metamodelo MOON. Esta plantilla consta de los siguientes apartados: nombre, descripción, motivación, entradas, precondiciones, acciones y postcondiciones.

Sin embargo, a pesar de a tener una definición precisa del metamodelo MOON, existía un cierto salto entre el metamodelo (expresado en UML), la definición de la refactorización (con texto y lógica de predicados de primer orden) y la implementación final (realizada con un lenguaje OO como JAVA).

Definición con Alf Aprovechando las propiedades de ALF como lenguaje de acciones sobre un modelo expresado en UML quedaría definida de la siguiente forma en ALF (ver Listado. 1).

Listado 1: Refactorización **Rename Class**

```
namespace Moon::Refactoring;

activity RenameClassRefactoring
  // Inputs
  (in classDef : ClassDef, in newName : Name) : Boolean
{
        // Preconditions
        if (!ExistsClassWithEqualName(classDef.nameSpace, newName)){
                // Actions
                RenameClassAction(classDef, newName);
        }
        // Postconditions
        return ExistsClassWithEqualName(classDef.nameSpace, newName);
}
```

Dicha refactorización depende del predicado `ExistsClassWithEqualName`, que puede ser expresado en ALF tal y como se detalla en el Listado. 2.

Listado 2: Predicado **ExistsClassWithEqualName**

```
namespace Moon::Predicate;

activity ExistsClassWithEqualName(in nameSpace: NameSpace, in newName : Name)
      : Boolean
{
    // exists a class in the namespace with equal name
        return nameSpace.classes -> exists c (c.getName() == newName);
}
```

Las modificaciones a las instancias del metamodelo se realizan a través del concepto de acción, modificando el estado de las instancias del metamodelo MOON (ver Listado. 3).

Listado 3: Acción **RenameClassAction**

```
namespace Moon::Action;

activity RenameClassAction
        (in classDef : ClassDef, in newName : Name)
{
  // set the new name of the class
  classDef.setName(newName);
}
```

5.2. Refatorización Push Down Field

Esta refactorización fue descrita en [13,14], y también está definida explícitamente en el catálogo de Fowler [1]. Su descripción detallada según [1] es la siguiente:

Nombre	**Push Down Field**
Motivación	Cuando un atributo es necesario en las subclases, pero no en las clase que lo define.
Mecanismos	- Declarar el atributo en todas las subclases. - Eliminar el atributo de la superclase. - Compilar y ejecutar tests. - Eliminar el atributo de todas las subclases que no lo necesitan. - Compilar y ejecutar tests.

A continuación se muestra la definición que se realizó en [13, Capítulo 5, Pág. 51]. La refactorización se denomina ***move_member_variable_to_subclasses***:

Nombre	**Move_member_variable_to_subclasses**
Descripción	Mover la variable miembro de su clase actual a cada una de las subclases.
Argumentos	variable (V), lista de subclases
Precondiciones	- La variable no es referenciada por los métodos de la clase, ni referenciada por instancias de la clase (se utiliza la consulta containingClass(V)): $referencesTo(V, containingClass(V)) = \emptyset$ - La variable está ya heredada por las subclases: $V.accessControlMode \neq private$.
Preservación del comportamiento	Una vez finalizada la refactorización, la variable V estará siempre definida en las subclases. Por lo tanto, la propiedad siete (equivalencia semántica en referencias y propiedades) es preservada por las precondiciones. El resto de propiedades se preservan.

Por su parte, en [14, Capítulo 3, Pág. 29] se denomina **PushDownInstanceVariable**, con una definición menos detallada:

PushDownInstanceVariable(class, varName) Elimina una variable de instancia de una clase y la añade en las subclases. Sólo es legal si no hay referencias a la variable en la clase.

Definición con Alf La refactorización quedaría definida de la siguiente forma en Alf (ver Listado. 4).

Listado 4: Refactorización **Push Down Field** utilizando Alf

```
namespace Moon::Refactoring;

// imports omitted

activity PushDownFieldRefactoring
    // Inputs
    (in attDec: Attribute) : Boolean
```

```
{

  // Pre:
  if ( ReferenceAttributeEmpty(attDec) &&
         !ExistsAttributeWithEqualNameInDirectDescendants(attDec)) {
        // Action
    // add the field in subclasses
        AddFieldInSubclassesAction(attDec);
    // remove the field in current class
        RemoveFieldAction(attDec);
  }
  // Post:
  // all descendants classes contains an attribute as attDec
  return ForAllDirectDescendantsIncludesAttributeAs(attDec);
}
```

Dicha refactorización depende de los predicados, `ReferenceAttributeEmpty`, `ForAllDirectDescendantsIncludesAttributeAs` y `ExistsAttributeWithEqualNameInDirectDescendants`. Por motivos de brevedad, se muestra en el Listado. 5 sólo uno de de dichos predicados en Alf.

Listado 5: Predicado **ReferenceAttributeEmpty**

```
namespace Moon::Predicate;

activity ReferenceAttributeEmpty(in attDec : Attribute) : Boolean
{
  // For all classes in clients
  // does not exist a reference to the attribute
  return attDec.classDef.getClients()
              -> forAll cd (cd.methods.body.getExpressions()
                    -> forAll ex (ex instanceof CallExpr
                              && ((CallExpr) ex).getFirstElement() != attDec));
}
```

Las modificaciones a las instancias del metamodelo se realizan a través del concepto de acciones: en este ejemplo, a través de las acciones `AddFieldInSubclassesAction` y `RemoveFieldAction`. En el Listado. 6 se muestra una de dichas acciones.

Listado 6: Acción **AddFieldInSubclassesAction**

```
namespace Moon::Action;

activity AddFieldInSubclassesAction(in attDec : Attribute) {
  // for each class in direct descendants
  // add an equivalent attribute
  for (ClassDef cd : attDec.classDef.getDirectDescendants()) {
    // create copy of initial attribute
        Attribute auxAttDec = new Attribute(attDec.getType(), attDec.getName
            ());
        cd.attributes -> add(auxAttDec);
  }
}
```

5.3. Ventajas y desventajas del uso de Alf

Los casos de estudio nos muestran algunas ventajas y desventajas en el uso de Alf, en contraposición a las soluciones presentadas previamente:

Ventajas

- El lenguaje permite expresar tanto el metamodelo (paquetes, clases, asociaciones, etc.) como las actividades que actúan sobre él, ya sea consultando o transformado la información, con un único lenguaje.
- La potencia del lenguaje para navegar sobre los modelos y utilizar funciones de expansión (*e.g.* `forAll`, `exists`, etc.) dotan de unas posibilidades similares a los cuantificadores de la lógica de predicados.
- Semántica precisa de la definición, no pudiendo darse diferentes interpretaciones una vez fijado el metamodelo y la refactorización. Este problema es acusado en catálogos con definiciones basadas en texto.
- Reducción del desajuste entre el modelo utilizado y la definición de la refactorización. En nuestro caso concreto, el metamodelo se expresa en UML, mientras que la refactorización se expresa en un lenguaje de acciones ideado para permitir la consulta y transformación de instancias de modelos UML.
- Reducción del desajuste entre la definición de la refactorización y la solución de implementación final. Mientras que la utilización de lógica de predicados dificultaba la expresividad en cuestiones relativas a las acciones, en Alf se pueden detallar las operaciones a realizar sobre las instancias del metamodelo, tanto desde una forma declarativa como también en forma imperativa.
- La traducción desde Alf a los distintos lenguajes de programación de la corriente principal, en particular aquellos orientados a objetos, se puede realizar de una forma suave, dado que el propio lenguaje ha sido diseñado con dicho objetivo.
- Validación del metamodelo para la definición de refactorizaciones, no se puede utilizar en Alf si el metamodelo no proporciona la información.
- Comprobación de la definición de una refactorización, con las herramientas asociadas a Alf se puede comprobar la sintaxis y la semántica estática.

Desventajas

- Falta de madurez en el desarrollo de herramientas que faciliten la transición de metamodelos como MOON a su correspondiente definición en Alf, aunque existen trabajos en curso [21].
- Actualmente, no existen herramientas para generar código ejecutable a partir de Alf. El compilador utilizado [18] realiza una comprobación sintáctica y de semántica estática, no genera código, pese a que este es uno de los objetivos finales del lenguaje de acciones. Esto impide poder realizar una comprobación de la preservación de la semántica dinámica.
- El hecho de que Alf proporcione una sintaxis tanto declarativa como imperativa, puede conducir a expresar las actividades en una forma condicionada a su implementación posterior en un lenguaje particular.
- La especificación de Alf está en una versión Beta en OMG, pendiente todavía de modificaciones. Aunque existen herramientas que ya están integrando Alf, pudiera estar todavía sometida a cambios.

6. Conclusiones y líneas de trabajo futuro

Desde el punto de vista de la adecuación de ALF como lenguaje para la definición de refactorizaciones a partir de una solución basada en metamodelos, podemos considerar que los resultados son adecuados, aún siendo conscientes de la necesidad de extender los resultados sobre un mayor conjunto de refactorizaciones del catálogo existente [1] así como validar con otras herramientas de desarrollo en curso [22].

Al emplear ALF se valida la adecuación y completitud del metamodelo MOON. A nuestro juicio esto puede ser extrapolado a otras soluciones basadas en metamodelos a la hora de abordar la independencia del lenguaje en refactorización.

La distancia entre la definición y la implementación se reduce, puesto que ALF surge precisamente para eliminar ese salto entre el modelado y su implementación ejecutable. La definición en ALF de la refactorización es traducible a un lenguaje de la corriente principal como JAVA o C#, y permite eliminar ambigüedades.

En la actualidad la traducción de ALF al lenguaje de implementación se realiza de forma manual. Sin embargo, algunos trabajos emergentes y herramientas en desarrollo [18,21] llevan a pensar que dicha distancia se reduzca en un futuro cercano, en cuanto estas herramientas alcancen un grado mayor de madurez.

Por otro lado, en este trabajo se han abordado refactorizaciones "clásicas". En la línea de otros trabajos ya presentados [19,23], surge la necesidad de validar en un futuro la definición de nuevas refatorizaciones.

Agradecimientos

Este trabajo ha sido financiado por el *Ministerio de Ciencia e Innovación*, en el proyecto TIN2008-05675.

Referencias

1. Martin Fowler. *Refactoring. Improving the Design of Existing Code.* Addison-Wesley, 1999.
2. Pieter Van Gorp, Hans Stenten, Tom Mens, and Serge Demeyer. Towards Automating Source-Consistent UML refactorings. In *UML*, pages 144–158, 2003.
3. Nabor C. Mendoça, Paulo Henrique M. Maia, Leonardo A. Fonseca, and Rossana M. C. Andrade. RefaX: A Refactoring Framework Based on XML. In *20th IEEE International Conference on Software Maintenance (ICSM'04)*, pages 147 – 156, 2004.
4. Ralf Lämmel. Towards Generic Refactoring. In *Proc. of Third ACM SIGPLAN Workshop on Rule-Based Programming RULE'02*, Pittsburgh, USA, October5 2002. ACM Press. 14 pages.
5. Serge Demeyer, Sander Tichelaar, and Patrick Steyaert. FAMIX 2.0 - the FAMOOS information exchange model. Technical report, Institute of Computer Science and Applied Mathematic. University of Bern, 1999.

6. Sander Tichelaar, Stéphane Ducasse, Serge Demeyer, and Oscar Nierstrasz. A Meta-Model for Language-Independent Refactoring. In *Proc. International Workshop on Principles of Software Evolution (IWPSE)*, pages 157–169. IEEE Computer Society Press, 2000.
7. Stéphane Ducasse, Michele Lanza, and Sander Tichelaar. Moose: an Extensible Language-Independent Environment for Reengineering Object-Oriented Systems. In *Proc. Int'l Symp. Constructing Software Engineering Tools (CoSET)*, June 2000.
8. Stéphane Ducasse, Michele Lanza, and Sander Tichelaar. The Moose Reengineering Environment. *Smalltalk Chronicles*, 2001.
9. Sander Tichelaar. *Modeling Object-Oriented Software for Reverse Engineering and Refactoring*. PhD thesis, University of Bern, 2001.
10. Pieter Van Gorp, Niels Van Eetvelde, and Dirk Janssens. Implementing Refactorings as Graph Rewrite Rules on a Platform Independent Meta model. In *Proceedings of 1st Fujaba Days*, october 2003.
11. Tom Mens, Niels Van Eetvelde, Serge Demeyer, and Dirk Janssens. Formalizing refactorings with graph transformationss. *J. Softw. Maint. Evol.*, 17(4):247–276, July 2005.
12. Yania Crespo. *Incremento del Potencial de Reutilización del Software mediante Refactorizaciones*. PhD thesis, Universidad de Valladolid, 2000. Available at http://giro.infor.uva.es/docpub/crespo-phd.ps.
13. William F. Opdyke. *Refactoring Object-Oriented Frameworks*. PhD thesis, University of Illinois at Urbana-Champaign, IL, USA, 1992.
14. Donald Bradley Roberts. *Practical Analysis for Refactoring*. PhD thesis, University of Illinois at Urbana-Champaign, IL, USA, 1999.
15. Bernhard Beckert, Uwe Keller, and Peter H. Schmitt. Translating the object constraint language into first-order predicate logic. In *Proceedings, VERIFY, Workshop at Federated Logic Conferences (FLoC*, pages 113–123, 2002.
16. OMG. Action Language for Foundational UML (Alf). Concrete Syntax for a UML Action Language - Beta 1, 2010.
17. OMG. Semantics of A Foundational Subset for Executable UML Models (fUML) version 1.0. www.omg.org/spec/FUM, 2011.
18. Data Access Technologies Inc. Action Language for UML (Alf) Parser. http://lib.modeldriven.org/MDLibrary/trunk/Applications/Alf-Reference-Implementation/dist/, 2012.
19. Raúl Marticorena, Yania Crespo, and Carlos López. Refactorizaciones en la Migración del Software. In *Actas JISBD'08, XIII Jornadas de Ingeniería del Software y Bases de Datos, Gijón, 2008. ISBN:978-84-612-5820-8*, pages 409–414, oct 2008.
20. Raúl Marticorena, Carlos López, Yania Crespo, and Francisco Javier Pérez. Refactoring Generics in JAVA: a case study on Extract Method. In *14th European Conference on Software Maintenance and Reengineering (CSMR)*, pages 217 – 226, April 2010.
21. CEA. Open Source Tool for UML modeling. http://www.papyrusuml.org/, 2012.
22. Elena Planas. alf-verifier - A lightweight tool for verifying UML-Alf executable models. http://code.google.com/a/eclipselabs.org/p/alf-verifier/, march 2012.
23. Raúl Marticorena, Carlos López, and Yania Crespo. Afrontando la Evolución de los Lenguajes de Programación a través de Refactorizaciones. In *Actas de la XVI Jornadas de Ingeniería del Software y Bases de Datos*, number ISBN: 978-84-9749-486-1, pages 277 – 290, La Coruña, 2011. Sistedes.

Usability mechanisms extension to ScrumTime

Ana M. Moreno, Agustin Yagüe, Diego Yucra

Universidad Politecnica de Madrid
Madrid, Spain
{ammoreno@fi.upm.es, agustin.yague@upm.es, diego.yucra@gmail.com}

Abstract. This contribution presents the extension of the existing ScrumTime tool to support usability mechanisms that has been done. The presented tool is used to manage agile projects. The extension increases the features of ScrumTime to define user stories through usability mechanisms selection, acceptance criteria definition and recommendation about usability tasks, acceptance criteria and usability stories. The tool is available to be tested.

Keywords: Agile development, usability patterns, user stories, HCI.

1. Motivation and background

During the last few years, the integration and cross pollination between usability and agile practices have been a rapidly expanding area of work and research. One of the premises of this line of work is that usability is a critical quality factor and needs to be dealt with during agile development to provide a quality user experience. Both the HCI and agile communities agree on this point. On the HCI side, for example, Nielsen [1] states that an agile development team must recognize interaction design and usability methods as explicit methodology development components, whereas, on the agile side, Ambler [2] claims that an end product's good usability can be ensured only by systematic usability engineering activities during development iterations.

There are many tools to support agile management projects either open source (Agilefant, IceScrum, Agilo, Scrumtime, …) or proprietary (Microsoft Team Foundation Server, IBM Jazz), but none has any specific features to deal with usability concerns.

Agile practices, addressed by the Agile Manifesto [3], pay special attention to human interaction, group management, customer collaboration and responding to change. Under this umbrella, the most of agile methodologies shares two common bases: iterative development and requirements specification using user stories. As a consequence of being iterative, the software product is developed in iterations tackling incremental functionality. Iterations consider either functional or non functional requirements. The agile artifact to represent product requirements is User Stories. A User story describes those products features that are expected by users concerning both functional and non functional issues[4]. User stories are characterized by three types of information: explanation (identification, name, description and priority), task enumeration and acceptance criteria. The task enumeration refers to activities that have to be done to carry out the user story.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 269-272, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

We have worked on the functional usability recommendations proposed in [5], that is, usability heuristics with key benefits (according to the usability literature) and with strong design implications, according to the software engineering literature. Table 1 provides an overview.

Table 1. Usability mechanisms addressed

Usability Mechanism	Goal
System Status	To inform users about the internal status of the system
Warning	To inform users of any action with important consequences
Long Action Feedback	To inform users that the system is processing an action that will take some time to complete
Global Undo	To undo system actions at several levels
Abort Operation	To cancel the execution of an action or the whole application
Abort Command	To cancel the execution of a task in progress
Go Back	To go back to a particular state in a command execution sequence
Structured Text Entry	To help prevent the user from making data input errors
Step-by-Step Execution	To help users to do tasks that require different steps with user input and correct such input
Preferences	To record each user's options for using system functions
Favorites	To record certain places of interest for the user
Multilevel Help	To provide different help levels for different users

Whether the sources of this information are customers/users, developers, usability experts or usability elicitation guidelines [5], such information may, from an agile perspective, require new user stories and/or modifications to the original functional stories (new acceptance criteria, new tasks…). We consider that this type of usability information should be somehow represented or documented as part of user stories, so they can be properly estimated and implemented.

2. Tool description

This section first presents the main features of ScrumTime before the extension, and the new ones that have been added to support usability mechanisms when creating user stories.

Fig. 1 Scrumtime dashboard.

Scrum Time is a web-based Scrum project management tool. This tool is intended to be intuitive, flexible, and extensible. It supports the following features: multi-project, backlog maintenance operations, sprint definition, project schedule and burn down charting. It is available at http://demo.scrumtime.com. User: demo Password: demo123. This version of ScrumTime has neither usability support or acceptance criteria to user stories. **Fig. 1** shows the tool dashboard.

To support the inclusion of usability mechanisms into user stories, we have modified an open source tool for managing user stories (ScrumTime http://www.scrumtime.org/). The resultant tool is available at http://scrumtime.eui.upm.es. The main features added to this new version of ScumTime are:

- The user story template has been completely re-structured to include the list of usability mechanisms available as checkbox items to be associated with each user story. Fig. 2 depicts the user story description template.

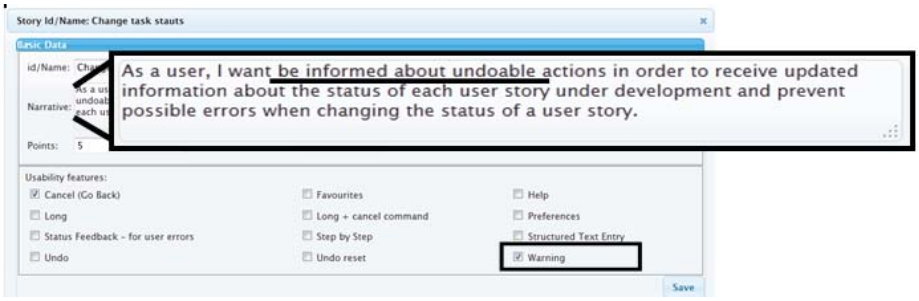


Fig. 2 User story description with usability features.

- A list of usability affected tasks: when a usability mechanism has been selected for a user story, recommendations about new tasks to be added (or the modifications to existing tasks) as a result of including this mechanism are displayed in the task panel as it is shown in Fig. 3.

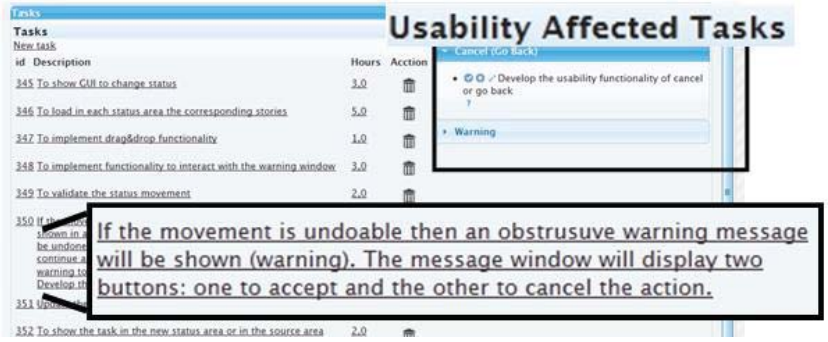


Fig. 3 User story tasks.

- A list of usability affected acceptance criteria: when a usability mechanism has been selected for a user story, the new criteria (or changes to existing criteria) to be taken into account to check that the implementation covers the usability features are displayed in the acceptance criteria panel. See Fig. 4.

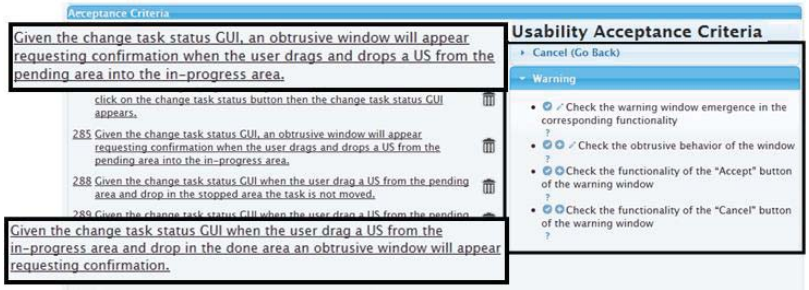


Fig. 4 User story acceptance criteria with usability.

- Usability story: when a usability mechanism has been selected for a user story and this mechanism requires the creation of a usability story, the usability story is automatically added to the product backlog. See Fig. 5

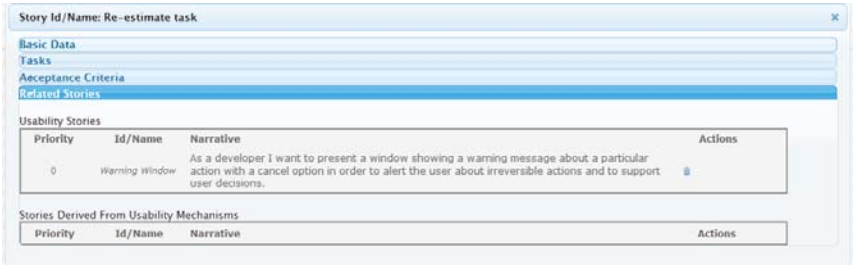


Fig. 5 Usability and related stories.

- Help functionality: examples on how to add usability tasks and acceptance criteria for each usability feature are provided through new help functionality.

The tool is under validation. It has been tested by UPM software engineering master students, all of whom have 2 to 4 years experience as software practitioners. As part of their master's thesis, they developed a real application using our tool for creating and documenting user stories. The description of the process to deal usability features using the tool is presented in [6].

3. Conclusion

We map the main usability mechanisms and their implications for user stories and also introduce a tool that captures the usability knowledge related to such implications. The approach is still undergoing validation, but preliminary results suggest that the workload for incorporating particular usability mechanisms using the stored usability knowledge leads is reasonably acceptable.

Acknowledgments. The work reported here has been partially sponsored by the Spanish Ministry of Industry, Tourism and Trade INNOSEP: INcorporating inNOvation in Software Engineering Processes project. (TIN2009-13849).

References

1. J. Nielsen, Agile Development Projects and Usability. In: Jakob Nielsen's Alertbox, November 17 (2008) http://www.useit.com/alertbox/agile-methods.html (Visited dec.2010)
2. S.W. Ambler, Tailoring Usability into Agile Software Development Projects. In: Law E., Hvannberg, E., Cockton G. (eds.) Maturing Usability. Quality in Software, Interaction and Value. Springer, Heidelberg (2008).
3. J. Nielsen, Usability Engineering. John Wiley & Sons, 1993.
4. M. vanWelie. Patterns in Interaction Design. http://www.welie.com. Accessed Nov. 2008.
5. Juristo N., Moreno A., Sanchez-Segura M-I.: Guidelines for eliciting usability functionalities. In: IEEE Trans. Softw. Eng., vol. 33, no. 11, pp. 744-758 (2007).
6. Moreno A., Yague A. Tailoring User Stories to Deal with Usability. JISBD 2012.

Tailoring user stories to deal with usability

Ana M. Moreno, Agustin Yagüe, Diego Yucra

Universidad Politecnica de Madrid
Madrid, Spain
{ammoreno@fi.upm.es, agustin.yague@upm.es, diego.yucra@gmail.com}

Abstract. Agile teams have to address usability to properly catch their users experience. But like in traditional software development, this task is not easy to achieve; there exists an interesting debate in the agile and usability communities about how to deal with this integration. In this paper we try to contribute to this debate by discussing the incorporation of particular usability recommendations into one of the most popular artifacts for communicating agile requirements, user stories. We discuss about the changes the incorporation of particular usability issues may introduce in a user story, and describe a tool that helps the agile team to deal with such usability issues during the specification of user stories. Some encouraging results about preliminary validation are also presented in the paper.

Keywords: Agile development, usability patterns, user stories, HCI.

1 Motivation

During the last few years, the integration and cross pollination between usability and agile practices have been a rapidly expanding area of work and research. One of the premises of this line of work is that usability is a critical quality factor and needs to be dealt with during agile development in order to provide a quality user experience. Both the Human Computer Interaction (HCI) and agile communities agree on this point. On the HCI side, for example, Nielsen [1] states that an agile development team must recognize interaction design and usability methods as explicit methodology development components, whereas, on the agile side, Ambler [2] claims that an end product's good usability can be ensured only by systematic usability engineering activities during development iterations.

This is not, however, a straightforward process. Different authors have highlighted challenges that need to be overcome if both fields want to work together. Differences in terminology (Ferreira et al. [3]), goals (Lee [4]) and approaches to software construction (Desilets [5]) are some of the most often cited obstacles to this integration.

Nonetheless, several topics dealing with this road to integration are under debate. At the organizational level, there is an interesting discussion about how the user experience (UX) team should work with the agile team (Ferreira et al. [6]). Another interesting line of work addresses when UX design should take place in an agile process (Constantine [7]) and Cooper [8], [9, 10, 11].

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 273-283, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

Some time ago, the HCI literature provided very specific usability recommendations with a clear positive impact on the final quality of use of software systems. Some examples are: give the user the option to cancel an ongoing process [9, 10, 11, 12], to undo a task [13, 14], provide the user feedback on what is going on in the system [13, 15, 16], adapt software functionalities to the user profile [17] or provide clear and marked exits for the application [15]. Such usability recommendations are in line with what Nielsen lately referred to as fast and cheap usability techniques [18], as quick usability actions that help to significantly increase user satisfaction.

Such recommendations represent specific functionalities to be incorporated into a software system. Therefore, as discussed by Juristo et al. in [19], they can be considered as functional usability requirements that complement traditional requirements.

Advancing along the above road to usability and agile integration, we address how to deal with the mentioned functional usability requirements in an agile context. We explore how to represent such functional usability requirements in user stories, one of the most popular artifacts for conveying agile requirements.

To do this, we have structured the paper as follows: Section 2 describes the usability recommendations that we will deal with. Then Section 3 discusses our proposal for documenting such usability recommendations into user stories. Section 4 introduces a software tool set up to support the inclusion of usability mechanisms in user stories. Section 5 describes preliminary results about the application of the previous approach in a case study. Finally, Section 6 outlines some conclusions and future work.

2 Functional Usability Requirements

We have worked on the functional usability recommendations proposed by Juriso et al. in [19], that is, usability heuristics with key benefits (according to the usability literature) and with strong design implications, according to the software engineering literature. Table 1 provides an overview.

Table 1. Usability mechanisms addressed

Usability Mechanism	Goal
System Status	To inform users about the internal status of the system
Warning	To inform users of any action with important consequences
Long Action Feedback	To inform users that the system is processing an action that will take some time to complete
Global Undo	To undo system actions at several levels
Abort Operation	To cancel the execution of an action or the whole application
Abort Command	To cancel the execution of a task in progress
Go Back	To go back to a particular state in a command execution sequence
Structured Text Entry	To help prevent the user from making data input errors
Step-by-Step Execution	To help users to do tasks that require different steps with user input and correct such input
Preferences	To record each user's options for using system functions
Favorites	To record certain places of interest for the user
Multilevel Help	To provide different help levels for different users

Notice that neither customers/users, nor, as Chamberlain et al. [11] claim, agile developers are generally usability experts. So, unless this type of usability information is documented in some way, good usability would, as Jokela and Abrahamsson [20] mentioned, be more or less a fluke resulting from customer and/or developer intuition. Whether the sources of this information are customers/users, developers, usability experts or usability elicitation guidelines [19], such information may, from an agile perspective, require new user stories and/or modifications to the original functional stories (new acceptance criteria, new tasks…). We consider that this type of usability information should be somehow represented or documented as part of user stories, so they can be properly estimated and implemented.

3 Specifying usability in user stories

Bearing in mind recommendations on how to write good user stories [21] and documentation on usability mechanisms [22], we have identified three ways in which the incorporation of usability influences user stories:

1. Addition of new stories to represent requirements directly derived from usability. We call these new stories "usability stories" to distinguish them from traditional user stories, as they represent usability features to be provided by the system.
2. Addition or modification of tasks in existing user stories. This means that some actions derived from usability constraints should be performed in an existing user story. This task could be as simple or detailed as needed.
3. Addition or modification of acceptance criteria. These acceptance criteria appear because the user story functionality needs to include some specific actions that modify the operating environment.

Based on [21, 23], the term usability story could be defined as "an artifact that is used to represent usability features that a system/software should support because they are needed by a user to use in a more easy and trusty way and that gives value to the user/acquirer. Usability stories are documented as user stories because both are similar. Next section shows an example of a usability story for implementing warning messages. User stories and usability stories are referred to differently to highlight that usability stories are created to address usability requirements related to a particular user story. Usability stories will be elicited not from product owners but from usability mechanisms designed to improve the use of a particular functionality represented in a user story. The next section gives an example of a user story including the warning usability mechanism.

Coming back to the implications of usability features into user stories, at least one of the previous three actions has to be taken when writing user stories with usability. Table 2 shows the implications of each analyzed usability mechanism when it is included in a user story. As we will see later it has been empirically validated. Table 2 columns represent the above actions and rows contain the usability mechanisms. Cells marked with an "X" signal that the incorporation of the usability mechanism requires the respective action. For example, the implementation of the warning mechanism

affects the user story by modifying acceptance criteria, adding new acceptance criteria, adding new tasks and adding a new usability story to the product backlog. Table 2 was built empirically as a result of two case studies and is being further validated, as discussed later.

Table 2. Mapping between usability mechanisms and actions

	New Task	Modify Task	New Acceptance Criteria	Modify Acceptance Criteria	New Usability Story	New User Story
System Status		X	X	X	X	
Warning	X		X	X	X	
Long Action		X	X		X	
Abort command	X	X	X		X	
Abort operation		X	X			
Go Back	X	X	X			
Text entry		X	X			
Step by Step	X		X	X		
Preferences						X
Favorites		X	X		X	
Help		X	X		X	

4 Tool support

To support the inclusion of usability mechanisms into user stories, we have modified an open source tool for managing user stories (ScrumTime http://www.scrumtime.org/). The resultant tool is available at http://scrumtime.eui.upm.es. There is available a guest account: guestdemo (password: scrumtime). The main features added to this new version of ScumTime are:

- A list of usability mechanisms available as checkbox items to be associated with each user story.
- A list of usability affected tasks: when a usability mechanism has been selected for a user story, recommendations about new tasks to be added (or the modifications to existing tasks) as a result of including this mechanism are displayed in the task panel.
- A list of usability affected acceptance criteria: when a usability mechanism has been selected for a user story, the new criteria (or changes to existing criteria) to be taken into account to check that the implementation covers the usability features are displayed in the acceptance criteria panel.
- Usability story: when a usability mechanism has been selected for a user story and this mechanism requires the creation of a usability story, the usability story is automatically added to the product backlog.
- Help functionality: examples on how to add usability tasks and acceptance criteria for each usability feature are provided through new help functionality.

Let us look at an example to illustrate how the tool works. Consider an application managing user stories in agile projects. One of the features of this application might

be "*graphically change the status (created, in progress, stopped, done) of a user story*". A user story description that does not consider usability features might read "As a user, I want to change the status of a user story and receive updated information about the status of each user story under development".

This user story description does not include any information about usability. Suppose, for example, that customers want to be warned about undoable actions (*warning feature*). The user story description could be written as follows: "As a user, I want to be informed about undoable actions in order to receive updated information about the status of each user story under development and prevent possible errors when changing the status of a user story." The underlined part of the description is derived from the warning need.

Following the process described by authors in [24], this feature should be added to the user story because some technical actions have to be taken to inform customers.

Fig. 1 shows how the tool does this. The description has been zoomed-in and the words related to the warning pattern have been highlighted.

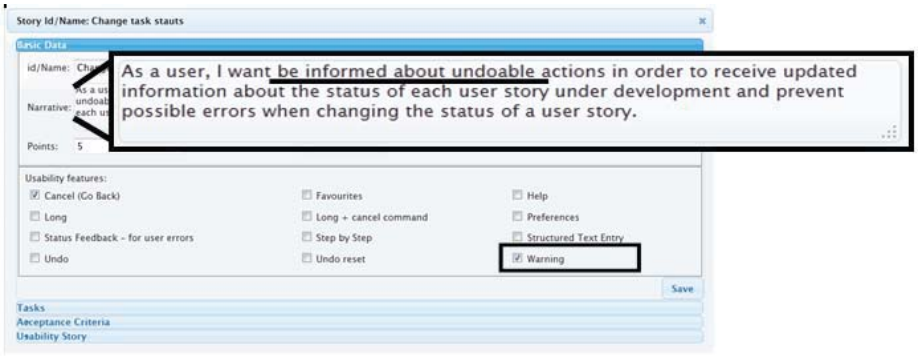


Fig. 1 User story description with usability features

After the user story is described and the usability features identified, the next is start with the conversation to know how is expected to work the user story. A candidate conversation is:

"*I want to select a user story dragging it from some area of the board and then move it and drop it to another area representing a change of status. When the movement means any undoable actions like to move a user story from not started to in-progress area, I want to be informed that the task time counter is going to start and that it is not possible to remove the accounted time. That is the reason why the movement is undoable.*"

Considering the conversation, the next step is to define the tasks and acceptance criteria that are necessary to implement the user story. Basic tasks and acceptance criteria are fixed later, when the user story is detailed during sprint planning. A new task has to be added to account for the usability feature. Some of the tasks are influenced by the warning need and some other are because of the functionality. Table 1 presents a possible task description for the user story. The light gray highlighted cells are the specific tasks that have been included to support the warning functionality.

Fig. 2 shows a screenshot of the modified version of ScrumTime with one of these tasks, highlighted by a zoomed-in black box, added to describe the warning task. All the tasks that are required because of the usability features are listed on the right side of the ScrumTime user interface. They are also boxed in black.

Table 2. User story tasks description

Description
To show GUI to change status
To load in each status area the corresponding stories
To implement drag&drop functionality
To implement functionality to interact with the warning window (added description)
To validate the status movement
If the movement is undoable then an obtrusive warning message will be shown (warning). The message window will display two buttons: one to accept and the other to cancel the action.
Update the server with the new US status
To show the task in the new status area or in the source area when illegal movements or system fails.

The last, but not the least, step is to identify and define the user story acceptance criteria. As it happened with tasks, some of the acceptance criteria come from usability issues. Table 3 shows a list of acceptance criteria that should be satisfied by the user story. Those acceptance criteria derived from usability issues are highlighted in color light grey.

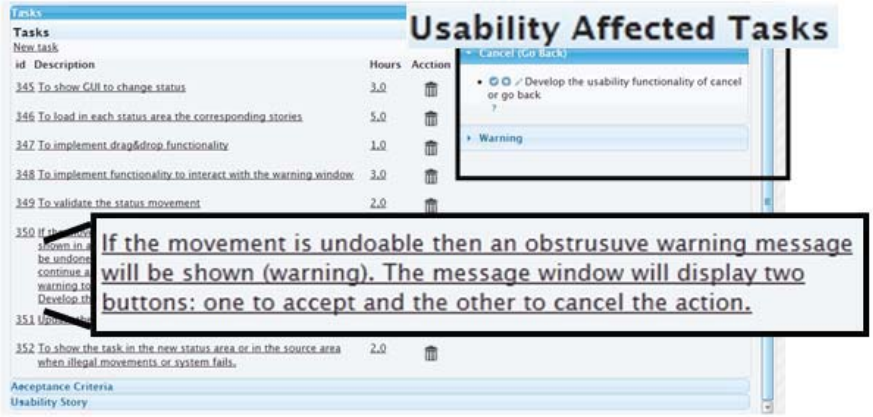


Fig. 2 User story tasks that support the warning feature

Table 3. User Story acceptance criteria

Descripción
Given the main agile project management application form when the user clicks on the change task status button then the change task status GUI appears.
Given the change task status GUI, an obtrusive window will appear requesting confirmation when the user drags and drops a US from the pending area into the in-progress area.
Given the change task status GUI when the user drags a US from the pending area and drops in the stopped area the task is not moved.
Given the change task status GUI when the user drags a US from the pending area and drops in the done area the task is not moved.
Given the change task status GUI when the user drags a US from the in-progress area and drops in the stop area the task is moved accordingly and the status information is updated in the server.
Given the change task status GUI when the user drags a US from the stop area and drops in the in-progress area the task is moved accordingly and the status information is updated in the server.
Given the change task status GUI when the user drags a US from the in-progress area and drops in the done area an obtrusive window will appear requesting confirmation.
Given the warning window, when the user clicks the Accept button to drop the US into the in-progress area, the task is moved according to the required action and the status information is updated in the server
Given the warning window and the user is dropping in the done area when the user clicks the cancel button the movement is not done and the task stays in the in-progress area.
Given the change task status GUI when the user drags a US from the done area and drops in the any other area the task is not moved.
Given the warning window and the user is dropping in the in-progress area when the user clicks the confirmation button the task is moved accordingly to the required movement and the status information is updated in the server. Check the functionality of the “Accept” button of the warning window
Given the warning window and the user is dropping in the in-progress area when the user clicks the cancel button the movement is not done and the task stays in the pending area. Check the functionality of the “Cancel” button of the warning window

Fig. 3 shows how these acceptance criteria are presented in ScrumTime. It is also needed a new usability story to represent the warning window itself.

As a direct consequence of the usability mechanisms that have been included in the user story, it is possible that new functionality should be added to the project as new user stories. These new user stories are called usability stories. In the previous example, it needed a user story to present a warning message. Most of the languages already provide this functionality, but in the opposite case, it is needed to implement.

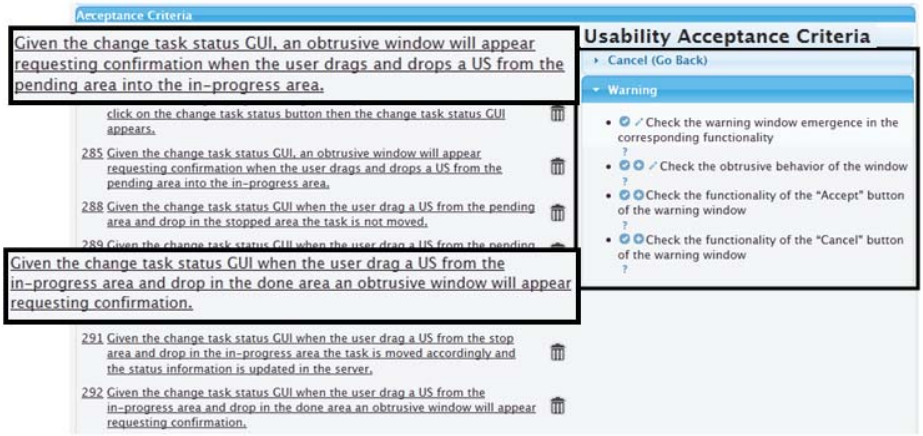


Fig. 3 User story acceptance criteria with usability

The description of the user story could be like:

"As a usability provider I want to present a window showing a warning message about a particular action with a cancel option in order to alert the user about irreversible actions and to support user decisions."

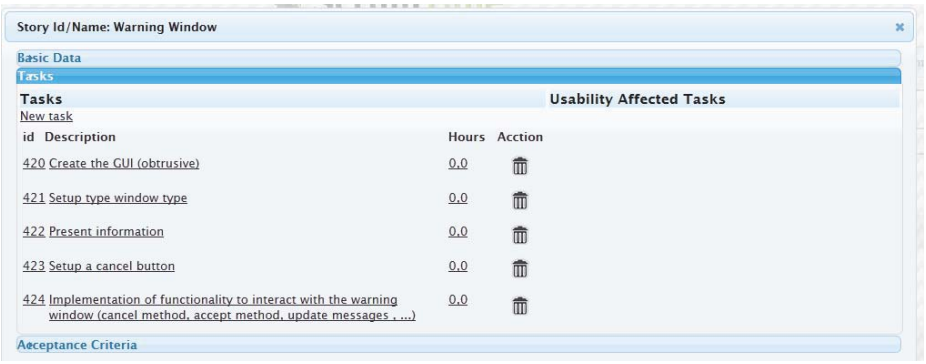


Fig. 4. Warning window usability story tasks.

Fig. 4 and **Fig. 5** show the tasks and acceptance criteria identified to describe a warning window.

If different usability mechanisms are associated with a user story, the implications for the tasks and acceptance criteria will appear in a compressed folder which users can expand at their convenience (Fig. 2 and **¡Error! No se encuentra el origen de la referencia.** show the go back and warning mechanisms, for example).

The main objective of the tool is to capture the usability meta-knowledge related to the inclusion of particular usability mechanisms in a software system. Consequently, a developer without too much usability knowledge can contribute to the development of usable systems. Notice that, as already discussed; neither users nor developers are

ordinarily usability experts. Therefore, an automated tool storing this usability meta-knowledge can be helpful if there are no usability experts on hand.

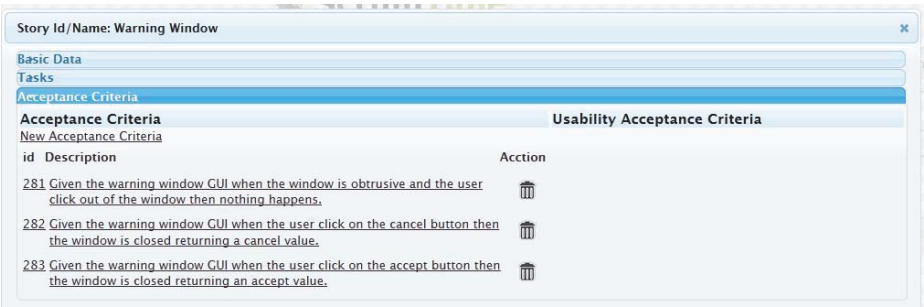


Fig. 5. Warning window usability story acceptance criteria

5. Preliminary validation

At the time of writing this paper, the validation process was still in progress. It is, however, a two-part process. First, we have worked on validating the usability knowledge summarized in Table 2. To do this, UPM software engineering graduate students developed a small agile project (a tool for managing user stories) as part of their degree project. The tool implemented 24 usability stories incorporating the described usability mechanisms. Using the results we were able to test the usability implications for tasks, acceptance criteria and new usability stories derived from each usability mechanism.

Second, we are validating the tool described in Section 4. It has been tested by UPM software engineering master students, all of whom have 2 to 4 years' experience as software practitioners. As part of their master's thesis, they developed a real application using our tool for creating and documenting user stories. In particular, we worked with three agile teams composed of 3 to 4 developers. The final results are still under evaluation, but early feedback suggests that usability features were easy to add to the user stories and not much usability knowledge was required to do so. The main identified problems were management issues concerning the removal of usability features after their tasks or acceptance criteria were defined.

Finally, from the preliminary experience using the tool, we can conclude that it is easy to check the usability features of each user story, but it takes some practice to incorporate the usability discussion into the regular user story creation flow. The tool is available at *http://scrumtime.eui.upm.es*.

6 Conclusion

Our hypothesis is that usability constraints may have a major impact on the system to be built. They should, therefore, be dealt with in the development process. This

paper aims to present preliminary results on incorporating particular usability mechanisms into agile user stories.

We map the main usability mechanisms and their implications for user stories and also introduce a tool that captures the usability knowledge related to such implications. The approach is still undergoing validation, but preliminary results suggest that the workload for incorporating particular usability mechanisms using the stored usability knowledge leads is reasonably acceptable.

The concept of usability story has been defined to represent the stories needed to implement the required usability mechanisms.

This research raises several issues, like, for example, when to deal with usability functionalities in an agile process or how to manage the size of user stories containing quite a few of usability mechanisms.

The next steps are related to further validating our solution and a detailed analysis of the open issues.

Acknowledgments. The work reported here has been partially sponsored by the Spanish Ministry of Industry, Tourism and Trade INNOSEP: INcorporating inNOvation in Software Engineering Processes project. (TIN2009-13849).

References

1. J. Nielsen, Agile Development Projects and Usability. In: Jakob Nielsen's Alertbox, November 17 (2008) http://www.useit.com/alertbox/agile-methods.html (Visited dec.2010)
2. S.W. Ambler, Tailoring Usability into Agile Software Development Projects. In: Law E., Hvannberg, E., Cockton G. (eds.) Maturing Usability. Quality in Software, Interaction and Value. Springer, Heidelberg (2008).
3. J. Ferreira, J. Noble and R. Biddle, Agile development iterations and UI design. In AGILE '07: Proc of the AGILE2007.Washington,DC,USA: IEEE Computer Society, pp.50-58.2007.
4. J.C. Lee, Embracing Agile Development of Usable Software Systems. In: CHI (2006)
5. A. Desilets, Are Agile Usability and Methodologies Comparable. In: http://www.carleton.ca/hotlab/hottopics/Articles/June2005-AreAgileandUxMet.html.Visited on Dec 2010, (2005).
6. Ferreira J., Sharp H., Robinson H. Values and Assumptions Shaping Agile Development and User Experience Design in Practice. In: Agile Processes in Sw Engineering and Extreme Programming. Proc. of 11th International Conference XP 2010, pp. 179-184 (2010),
7. Constantine L. L.: Process agility and software usability: Toward lightweight usage-centered design. Constantine & Lockwood, Ltd., Tech. Rep. 110 (2001). [Online]. Available: http://citeseer.ist.psu.edu/465732.html
8. Miller L.: Case study of customer input for a successful product. In ADC '05: Proceedings of the Agile Development Conference. Washington, DC, USA: IEEE Computer Society, pp. 225-234 (2005).
9. Patton J.: Hitting the Target: Adding Interaction Design to Agile Software Development. In: OPSLA'04 Proceedings (2004).
10. Haikara J.: Usability in agile software development: Extending the interaction design process with personas approach. In XP, 2007, pp. 153-156 (2007)
11. N. M. Stephanie Chamberlain, Helen Sharp, Towards a Framework for Integrating Agile Development and User-Centred Design. Springer Berlin. Heidelberg, 2006.
12. Usability Pattern Collection. http://www.cmis.brighton.ac.uk/research/patterns/ (Dec 2010).

13. B. Shneiderman. Designing the User Interface: Strategies for Effective Human-Computer Interaction. Addison-Wesley, 1998.
14. J. Tidwell. Designing Interfaces. Patterns for Effective Interaction Design. OReilly, 2005.
15. J. Nielsen, Usability Engineering. John Wiley & Sons, 1993.
16. M. vanWelie. Patterns in Interaction Design. http://www.welie.com. Accessed Nov. 2008.
17. Rubinstein R, Hersh H.: The Human Factor. Digital Press, Bedford, MA (1984).
18. Nielsen J.: Fast, Cheap, and Good: Yes, You Can Have It All. January (2007)) http://www.useit.com/alertbox/fast-methods.html (Visited December, 2010)
19. Juristo N., Moreno A., Sanchez-Segura M-I.: Guidelines for eliciting usability functionalities. In: IEEE Trans. Softw. Eng., vol. 33, no. 11, pp. 744-758 (2007).
20. Jokela T., Abrahamsson P.: Usability Assessment of an Extreme Programming Project. Close Co-operation with the Customer Does Not Equal Good Usability. In: PROFES (2004).
21. M. Cohn, User Stories Applied: For Agile Software Development (The Addison-Wesley Signature Series). Addison-Wesley Professional, March 2004. [On-line]. Available: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20n&path=ASIN/0321205685
22. N. Juristo, A. M. Moreno, and M.-I. Sanchez-Segura, Analysing the impact of usability on software design, J. Syst. Softw., vol. 80, no. 9, pp. 1506-1516, 2007.
23. Beck, K.: Extreme Programming Explained: Embrace Change. Addison Wesley (1999)
24. Moreno A.M., Yague A.: Adding usability recommendations into Agile user stories. Proc 1st workshop Dealing with Usability in an Agile Domain at XP 2010. (2010)

A. Ruíz, L. Iribarne (Eds.): Actas de las "*XVII Jornadas de Ingeniería del Software y Bases de Datos (JISBD'2012)*", Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

Assessing the influence of stereotypes on the comprehension of UML sequence diagrams: A family of experiments

José A. Cruz-Lemus[1], Marcela Genero[1], Danilo Caivano[2], Silvia Abrahão[3], Emilio Insfrán[3], and José A. Carsí[3]

[1]Department of Technologies and Information Systems, University of Castilla-La Mancha, Spain
{JoseAntonio.Cruz, Marcela.Genero}@uclm.es
[2]Department of Informatics, University of Bari, Italy
caivano@di.uniba.it
[3]Department of Information Systems and Computation, Universidad Politécnica de Valencia, Spain
{sabrahao, einsfran, pcarsi}@dsic.upv.es

1 Introduction

Stereotypes are often used in industrial contexts and their application spans from use cases to class diagrams. Indeed, companies use stereotypes within their development processes to specialize general processes aiming to fit them to a particular technology in use, such as programming languages (e.g. C#, Java), application type (e.g. real-time, Web applications, client-server, standalone), reusable component used (e.g. Microsoft Foundation Class Library, Enterprise Java Beans Library) or simply to give more detailed guidelines to the practitioners involved in the system development processes.

Nevertheless, the influence of stereotypes on the comprehension of requirements models, such as UML sequence diagrams, had not been investigated yet. This fact motivated us to develop the research presented in this work.

2 Stereotypes and UML sequence diagrams

Sequence diagrams are a means to model an aspect of the dynamic behavior of a system and can be attached to a use case or to an object service in order to describe its expected behavior. This work presents the empirical validation of four stereotypes for UML sequence diagrams. These stereotypes were proposed to improve the comprehension of sequence diagrams attached to use cases according to the nature of the interactions (i.e. message interchange among participating object types). The four proposed stereotypes are:

- **<<signal>>** which is used with messages that represent interactions between actors and the system interface,

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 285-286, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

- **<<service>>** which is used with messages that represent the change of the internal state of an object of the receiving object type,
- **<<query>>** which is used with messages that represent queries about other objects or about class population, and
- **<<connect>>** which is used with messages that capture the need of a structural relationship among participating object types.

3 The family of experiments

Fig. 1 presents the road-map of the family of experiments carried out to verify whether the previously introduced stereotypes improve the comprehension of UML sequence diagrams. It includes information about number of subjects, materials language, location and date of the empirical study.

Experiment	Replica 1	Replica 2
# students: 78 (Computer Science - 4th year) *Language:* italian *Location:* University of Bari (Italy) *Date:* February 2008	*# students:* 29 (Computer Science – 3rd year) *Language:* italian *Location:* University of Castilla-La Mancha (Spain) *Date:* April 2008	*# students:* 36 (Computer Science - 5th year) Language: Spanish *Location:* University of Castilla-La Mancha (Spain) *Date:* April 2008

Fig. 1. Family of experiments road-map

The comprehension of the UML sequence diagrams was evaluated from three different perspectives borrowed from the Cognitive Theory of Multimedia Learning: *semantic comprehension* (ability to comprehend the semantics of the models), *retention* (comprehension of material being presented, and the ability to retain knowledge from it), and *transfer* (ability to use the knowledge gained from the material to solve related problems which are not directly answerable from it).

4 Conclusions

The obtained results indicated that using the presented stereotypes improves the comprehension of UML sequence diagrams, especially when the domain of the system that is being modeled is not well-known. Nevertheless, the magnitude of the results was not statistically significant enough to be generalized. Actually, the hypothesis tests carried out and the meta-analyses done show a positive effect of the proposed stereotypes but, at the same time, the size of this effect is too small to assume these results to be definitive.

References

1. J.A. Cruz-Lemus, M. Genero, D. Caivano, S. Abrahão, E. Insfrán, and J.A. Carsí. Assessing the influence of stereotypes on the comprehension of UML sequence diagrams: A family of experiments. Information and Software Technology 53(12), 1391-1403 (2011)

Metaheurísticas como soporte a la selección de requisitos del software

Isabel M. del Águila, José del Sagrado, and Francisco J. Orellana

Dpto. Lenguajes y Computación,
Ctra Sacramento s/n, 04120 Universidad de Almería, Espańa
{imaguila,jsagrado,fjorella}@ual.es

Resumen Las técnicas de optimización y metaheurísticas han sido aplicadas ampliamente en numerosas áreas, entre ellas la Ingeniería del Software. En este trabajo mostramos la incorporación de estas técnicas como soporte a las tareas de selección de un grupo de requisitos de entre aquellos que han sido propuestos por los clientes, validando experimentalmente sus resultados. Los algoritmos metaheurísticos son ejecutados desde una herramienta web que permite la definición colaborativa de los requisitos de un proyecto software y ayudan a los desarrolladores durante la ejecución del mismo.

Keywords: gestión de requisitos, ingeniería de requisitos asistida por computadora, optimización metaheurística

1. Introducción

La aplicación de técnicas de Inteligencia Artificial (IA) en los procesos de desarrollo de software ha obtenido buenos resultados en aquellas actividades que necesitan utilizar conocimiento experto. La etapa relativa a los requisitos de un proyecto software es una buena candidata a la aplicación de estas técnicas, debido a la naturaleza de los requisitos, que tienden a ser imprecisos, incompletos y ambiguos [10]. Además, numerosos estudios estadísticos, como los informes CHAOS publicados desde 1.994, indican que las tareas relacionadas con los requisitos son la principal causa de desastre de la puesta en valor del software [16,1,20]. Así, la utilización de técnicas de IA en la mejora del flujo de trabajo de los requisitos, afectará favorablemente a todo el ciclo de vida del desarrollo de software.

En este trabajo mostramos como aplicar con éxito algunas de estas técnicas a tareas relacionadas con los requisitos, en concreto la priorización y selección de requisitos, utilizando técnicas de optimización combinatoria. Esta solución instancia un patrón arquitectónico que permite la integración sin costuras de procesos de Ingeniería de Requisitos con técnicas de AI sobre la herramienta CARE (Computer Aided Requirement Engineering) InSCo-Requisite [25]. Esta herramienta de carácter académico ha sido desarrollada por el grupo de investigación (DKSE – Ingeniería de Datos, del Conocimiento y del Software) en la

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 287-297, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

Universidad de Almería proporcionando al grupo de desarrolladores y usuarios un entorno en el que trabajar de forma colaborativa en la definición de requisitos [21].

El resto del artículo se estructura como sigue. La sección 2 enmarca el trabajo en el ámbito de los procesos de Ingeniería de Requisitos (InRe). El problema de selección de requisitos se formaliza en la sección 3, para mostrar los enfoques de solución en la sección 4 donde también se resumen los resultados experimentales de la aplicación de las metaheurísticas. El resultado de su incorporación en InSCo-Requisite se muestra en la sección 5 para finalmente, en la sección 6, presentar las conclusiones.

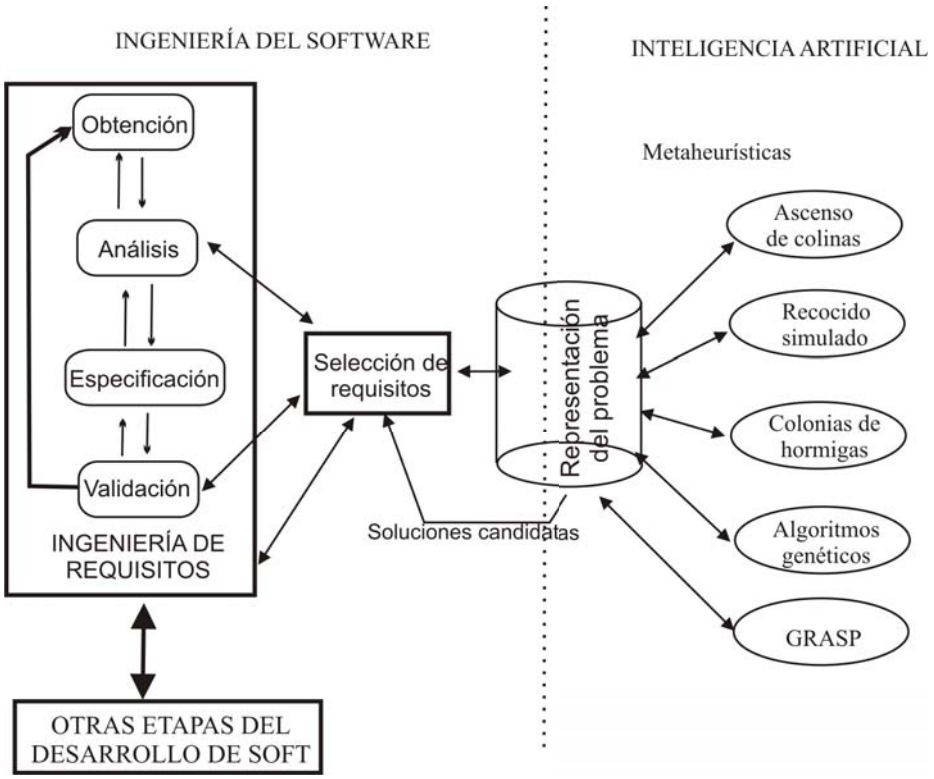


Figura 1. Sinergia entre InRe e IA

2. Visión general de la Ingeniería de Requisitos

La figura 1 muestra la versión más simple del flujo de trabajo de los requisitos [3]. Los requisitos se obtienen de los usuarios o cualquier persona implicada

en la definición del proyecto. Después se estudian y analizan para detectar inconsistencias o incompletitudes, generando una especificación en un determinado lenguaje. Las herramientas CARE ofrecen el entorno para utilizar bases de datos de requisitos, dando soporte a las tareas de especificación de requisitos en formato electrónico y permitiendo la gestión efectiva del proyecto. La validación permite comprobar los requisitos dando integridad al resultado de esta etapa del desarrollo. El ciclo elicitación-análisis-validación se ejecuta durante tantas iteraciones como sea necesario hasta que la especificación de requisitos (SRS-Software Requirement Specification) [2], se haya completado.

La selección de requisitos, de entre los que se hayan definido en los procesos de la InRe, es una tarea importante dentro de esta etapa. Puede aparecer durante el análisis, en las labores de negociación con los clientes o al final del ciclo de requisitos para fijar el alcance del producto a construir. Pero si se utilizan métodos de desarrollo iterativos, en especial los métodos ágiles [7,26], la selección de requisitos es una tarea básica que se retoma a inicio de cada iteración; en este punto, sin opción a cambio durante la ejecución de la iteración, se fijan los requisitos funcionales que se abordan en ese ciclo ágil. Por lo tanto la selección de requisitos adquiere una mayor relevancia dentro de todo el ciclo de desarrollo de software, ya que está presente al principio de cada iteración. Este problema es un proceso de toma de decisiones que utiliza conocimiento experto; el modelado e incorporación de este conocimiento a una herramienta CARE supone un avance importante en el desarrollo de software en general. El hecho de poder hacer cambios fácilmente en InSCo-Requisite es una oportunidad excepcional de poner en marcha la integración de la Ingeniería de los Requisitos (InRe) con las técnicas de IA, haciendo que ambas disciplinas mantengan su independencia [25].

3. Formalización del problema de selección de requisitos

Extrapolando el vocabulario médico, la selección de un conjunto de requisitos, se puede llamar triaje de requisitos [11], y se define como: *el proceso de determinar que requisitos debe satisfacer un producto software según los recursos y el tiempo disponible.* El software, debido a su naturaleza lógica es un complejo producto industrial, y suele venir definido por un gran número de requisitos que lo caracterizan y que en la mayoría de los casos no pueden completarse dentro de las restricciones definidas por tiempo y recursos, y que, por tanto, deben limitarse de alguna manera [8]. La solución es priorizar estos requisitos candidatos y seleccionarlos según los recursos disponibles. Debido al alto número de posibles soluciones, este problema se ha formulado y resuelto empleando técnicas de búsqueda [5,6,24,13,15,18,28,27]

A continuación proponemos la formulación completa de este problema ya que, tal como se muestra en la figura 1, será el punto de conexión entre las técnicas de búsqueda y las herramientas CARE. InSCo-Requisite trasformará el modelo de los requisitos que guarda en su base de datos en una representación que puedan manejar los algoritmos de búsqueda, y a su vez, estos algoritmos como salida, marcarán los requisitos que formarán parte de su solución.

Sea $R = \{r_1, r_2, \ldots, r_n\}$ el conjunto de requisitos propuestos por un conjunto de clientes $C = \{c_1, c_2, \ldots, c_m\}$. Los clientes tienen asignado un peso w_i que mide su importancia dentro del proyecto. Cada requisito $r_j \in R$ tienen asociado un esfuerzo de desarrollo que representa los recursos necesarios para incluir ese requisito en el producto software final, $E = \{e_1, e_2, \ldots, e_n\}$.

Cada $r_j \in R$ puede ser sugerido por más de un cliente y cada cliente c_i asigna un valor v_{ij} al requisito que representa la importancia que para él tiene la inclusión de ese requisito. Es decir, el valor representa la importancia del requisito r_j para el cliente c_i. Estos valores forman una matriz $m \times n$. La satisfacción global, s_j, o el valor añadido de la inclusión del requisito r_j en el producto software, se puede calcular como la suma ponderada de los valores de importancia para cada cliente, $s_j = \sum_i^m w_i v_{ij}$. Mayores valores de v_{ij}, implican una mayor prioridad de ese requisito para el cliente c_i. Un valor cero representa que ese cliente no toma en consideración ese requisito.

Habitualmente los requisitos están interrelacionados o presentan dependencias, que hacen que tengan que abordase unos antes que otros o que sean excluyentes. Las interacciones entre los requisitos representan restricciones al problema y se agrupan en dos tipos: dependencias funcionales o estructurales que representan relaciones semánticas que no pueden ser ignoradas ni relegadas a un tratamiento posterior en ningún caso, y dependencias por recursos consumidos, que suponen reajustar los valores de esfuerzo o satisfacción por la presencia conjunta o no de dos o más requisitos durante el proceso de selección. Aunando las dos grandes aproximaciones a la definición de las dependencias funcionales [9,4] podemos definirlas como:

- *Implicación.* (r_i *implica* r_j). El requisito r_i no puede ser seleccionado si el requisito r_j no ha sido implementado previamente.
- *Combinación.* (r_i *acoplado-con* r_j). El requisito r_i no puede ser incluido separadamente del r_j.
- *Exclusión.* (r_i *excluye-a* r_j). El requisito r_i no puede incluirse junto al requisito r_j.

De esta forma el objetivo del problema será encontrar el subconjunto de requisitos $\hat{R}$ de entre todos los subconjuntos de R, $\hat{R} \in \wp(R)$, que sea la mejor solución. La calidad de la solución se mide respecto a uno o varios objetivos utilizando una función de evaluación previamente fijada. Generalmente, en el dominio del problema de la selección de requisitos, se persigue construir $\hat{R}$ de forma que satisfaga en la mayor medida las peticiones de los clientes dentro de las limitaciones ofrecidas por los recursos del proyecto y las interacciones entre los requisitos. En términos de la formulación realizada para el problema esto lo conseguimos maximizando la satisfacción y dentro de los límites de esfuerzo. La satisfacción y el esfuerzo del conjunto $\hat{R}$ son respectivamente:

$$\text{sat}(\hat{R}) = \sum_{j \in \hat{R}} (s_j), \quad \text{eff}(\hat{R}) = \sum_{j \in \hat{R}} (e_j) \tag{1}$$

donde j representa al requisito r_j. Siendo B el límite de esfuerzo. Formalmente,

$$\begin{array}{l} \text{maximize sat}(\hat{R}) \\ \text{subject to eff}(\hat{R}) \leq B \end{array} \tag{2}$$

4. Solución basada en metaheurísticas

Las técnicas de optimización y metaheurísticas han sido ampliamente aplicadas en numerosas áreas de la ingeniería y como no podría ser de otra forma la ingeniería del software es una de ellas. Esta sinergia está siendo explotada por una corriente de investigación conocida como Ingeniería del Software Basada en Búsqueda (Search Based Software Engineering) [19]. En concreto en relación con la especificación y los requisitos nos encontramos con un gran número de trabajos que se centran el problema de la selección de los requisitos para la siguiente versión del software (Next Release Problem, NRP [5]). Un resumen de las más significativas se muestra en la tabla 1

Tabla 1. Trabajos que solucionan NRP

	Con interacciones	Sin interacciones
Recocido simulado	Bagnall et al. [5]	Baker et al. [6], del Sagrado et al. [24]
Alg. Voraz (Greedy)	Bagnall et al. [5]	Baker et al. [6]
GRASP	del Sagrado et al. [23]	
GA	Greer & Ruhe [18]	del Sagrado et al. [24]
NSGA-II	Zhang & Harman [27]	Durillo et al. [13], Finkelstein et al. [15], Zhang et al. [28]
MOCell		Durillo et al. [13]
ACS	del Sagrado et al. [23]	del Sagrado et al. [24]

Las técnicas metaheurísticas utilizadas para resolver NRP son muy diversas. Bagnall et al. [5] utilizó en su resolución el ascenso de colinas, una implementación de algoritmos voraces y la técnica del recocido simulado. El trabajo de Baker et al. [6] demostró la aplicabilidad de estas técnicas en casos reales mejorando la efectividad de los expertos. Greer y Ruhe [18], abordaron el problema como un problema de planificación, asignando los requisitos a incrementos, considerando las valoraciones de los clientes y las limitaciones de recursos. Como base para la optimización se utilizó un algoritmo genético. Del Sagrado et al. [24] adaptaron un sistema de colonia de hormigas a NRP, comparando sus soluciones con un algoritmo genético y el recocido simulado. NRP también ha sido tratado como un problema multiobjetivo, aplicando técnicas como NSGA−II, MoCell [28,13,15].

Sólo algunos de estos trabajos han tratado las interacciones entre los requisitos [5,18,27,23]. Bagnall et al. [5] sólo tiene en cuenta las dependencias de impliación y selecciona clientes, con todas sus propuestas, no selecciona requisitos. Greer y Ruhe [18] definen incrementos utilizando las interacciones de implicación y

combinación. Un estudio de cómo afectan las dependencias a los algoritmos genéticos aparece en [27]. El primer trabajo en donde se presentó una representación explicita de los requisitos mediante un grafo de interacciones es [23].

Una metaheuristica ofrece un estrategia para el diseño de heurísticas que den solución a un problema de optimización. Una de las principales ventajas es que permiten obtener soluciones cercanas a las óptimas empleando tiempos de ejecución razonables, gracias principalmente a una exploración más eficiente del espacio de búsqueda. La aplicación de una técnica metaheurística necesita, además de la representación del problema ya descrita, la definición de la estrategia de búsqueda de la solución, específica para cada técnica.

En este trabajo hemos seleccionado tres técnicas para resolver NRP: greedy randomized adaptive search procedure (GRASP), algoritmos genéticos y un sistema de colonia de hormigas (ACS), mostrando su adaptación al problema y una breve comparativa de los resultados que ofrecen sobre dos conjuntos de datos.

GRASP [14], aplica un esquema iterativo, primero construye una solución voraz aleatoria que se mejora con búsqueda local. Se construye una lista de elementos ordenados en base a una función voraz y se realiza una selección aleatoria. La lista se asegura de cumplir las restricciones del problema en cuanto a interacciones y límites de recursos. La función voraz mide la productividad del requisito.

$$g(r_i) = \frac{s_i}{e_i} \tag{3}$$

La búsqueda local intenta mejorar la solución actual buscando en el vecindario. Así, cosiderando las restricciones del problema, intenta sustituir cada requisito por otro que mejore la función de evaluación de la solución, $sat(R')$. El algoritmo acaba cuando se han explorado todos los requisitos de la solución.

Un algoritmo genético emula la evolución natural de una población o conjunto de individuos [17]. Durante una generación cada nueva población se construye aplicando operadores de selección, cruce y mutación sobre los individuos. Los individuos representan posibles soluciones, es decir, un conjunto de requisitos que satisfacen NRP. Los mejores individuos son los que tienen más posibilidades de ser elegidos como padres. Así un individuo $R_k \in \wp(R)$, de la población actual Q, es elegido como padre con una probabilidad:

$$p_{R_i} = \frac{sat(R_i)}{\sum_{R' \in Q} sat(R')} \tag{4}$$

La optimización por colonia de hormigas (Ant Colony Optimization, ACO) es una técnica de búsqueda que emula el comportamiento de las hormigas cuando busca el camino más corto desde su hormiguero hasta la comida [12]. Este proceso natural utiliza una sustancia química llamada feromona que segregan las hormigas conforme avanzan. Al seleccionar un camino una hormiga adopta un comportamiento probabilístico. En cada encrucijada, tiende a seleccionar el camino que tiene más feromona, si la cantidad de esta sustancia no es suficiente

para hacer una elección se hace de forma aleatoria. Conforme pasa el tiempo, los caminos más frecuentados tendrán una mayor concentración de feromona.

Los pasos de la ejecución de este tipo de algoritmos son:

- Inicialización del rastro de feromona. Representado por una matriz que recoge la cantidad de feromona inicial que se actualizará durante la ejecución del algoritmo.
- Selección aleatoria del punto de inicio de cada hormiga.
- En cada etapa la hormiga localiza los nodos visibles, en función de las restricciones del problema, y elige uno combinando la información heurística y la feromona.
- Actualización de la matriz de feromona de la mejor solución. La feromona de un camino (arco) del nodo i al nodo j se denota τ_{ij}.

Cuando se construye la solución de la hormiga k, la regla utilizada para seleccionar desde el nodo i el nodo j se define como:

$$j = \begin{cases} argmax_{u \in N_i^k}[[\tau_{ij}]^\alpha [\eta_{ij}]^\beta], \text{ if } q \leq q_0, \\ u, \qquad\qquad\qquad\qquad \text{otherwise.} \end{cases} \tag{5}$$

donde q es un valor aleatorio entre $[0, 1]$, y $q_0 \in [0, 1]$, con $(q \leq q_0)$ es un parámetro que controla el balanceo entre intensificación y diversificación en el algoritmo, buscando la mejor cobertura del espacio de búsqueda. La probabilidad con que la hormiga k estando en el nodo i seleccione moverse al j es:

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{h \in N_i^k} [\tau_{ij}]^\alpha [\eta_{ij}]^\beta}, \text{ if } j \in N_i^k, \\ 0, \qquad\qquad\qquad\qquad \text{otherwise.} \end{cases} \tag{6}$$

siendo la información heurística una media de productividad $\eta_{ij}^k = \mu(s_j/e_j($ con μ un valor de normalización y N_i^k el conjunto de nodos visibles para la hormiga k según las restricciones del problema. La mejor solución encontrada por las hormigas, $\hat{R}$, actualiza el rastro de feromona de acuerdo con:

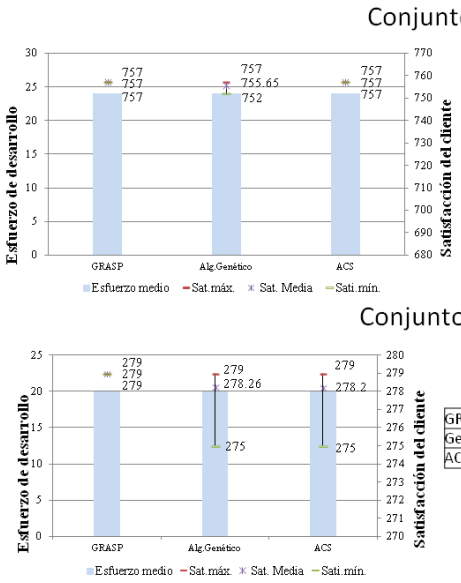
$$\tau_{ij} = (1 - \rho) * \tau_{ij} + \rho \Delta \tau_{ij} \tag{7}$$

siendo

$$\Delta \tau_{ij} = \frac{sat(\hat{R})}{sat(R)} \tag{8}$$

y $\rho \in [0, 1]$ la tasa de evaporación de la feromona,

Estos algoritmos han sido probados empleando dos conjuntos de datos, el primero de 20 requisitos y 5 clientes con $B = 25$, usado por primera vez en [6] y que ha sido ampliamente utilizado para evaluar este tipo soluciones al problema de selección de requisitos [27,13,23], y un segundo conjunto de datos sintético de 100 requisitos, 5 clientes y con $B = 20$. Sobre cada algoritmo se han realizado 100 ejecuciones independientes y sus resultados se muestran en las tablas. Comprobamos que en todos los casos se obtiene el mayor esfuerzo, y que todos los algoritmos alcanza un en algún caso el valor máximo de satisfacción.



	Satisfacción	Esfuerzo	Tiempo (ms)
GRASP	757±0	24	94.14±6.84
Genético	755.65±2.23	24	201.32±33.93
ACS	757±0	24	232.84±36.21

	Satisfacción	Esfuerzo	Tiempo (ms)
GRASP	279±0	20	1702.53±89.6
Genético	278.26±1.01	20	9744.1±229.9
ACS	278.2±1.13	20	20670.24±6235.6

Figura 2. Resultados de las metaherurísticas

Las variaciones del algoritmo genético y la colonia de hormigas, indican no se ha obtenido la convergencia deseada en algunos casos, (en especial en el conjunto de datos de mayor número de requisitos), pero como conclusión general de la experimentación se comprueba que con los tres métodos metaheurísticos se pueden obtener soluciones válidas para los ingenieros del software.

5. Selección de requisitos en InSCo-Requisite

En este y otros trabajos se ha mostrado de forma experimental la validez de las técnicas metaheurísticas en la solución del problema de selección de requisitos. Sin embargo la validez de estas técnicas está ligada a su utilización real insertadas en herramientas necesarias para el desarrollador de software y facilitan el trabajo cotidiano de estos profesionales tal como puso de manifiesto Ian Sommerville en las sesiones invitadas del primer simposio internacional SBSE [22]. El valor real de estos enfoques es combinar la inteligencia computacional con la experiencia de los expertos, en este caso ingenieros del software, para obtener mejores resultados del que obtendrían sólo los expertos. Es decir, GRASP, algoritmos genéticos y ACS, tienen que embeberse dentro de las herramientas software empleadas en el proyecto.

InSCo-Requisite facilita la obtención de los requisitos de los clientes [21] en una serie de proyectos. En cada proyecto participan un conjunto de clientes

dados previamente de alta en el sistema; al asignarlos a un proyecto se les fija un peso w_j, que puede variar entre proyectos, dependiendo de la importancia de ese usuario en cada proyecto concreto.

Esta aplicación web ofrece un conjunto de plantillas que permiten definir *requisitos funcionales, no funcionales y de información*, estos últimos son aquellos relativos a los datos manejados por el software a construir. Estos tres tipos de requisitos son los de más bajo nivel, ya que InSCo-Requisite mantiene una estructura jerárquica que permite manejar diversos niveles de abstracción. Así, se representan de forma separada los *objetivos*, relacionados con el nivel de negocio que se refinan hasta llegar a *requisitos funcionales*, y lo mismo con los *conceptos del dominio* y los *requisitos de información*. Los requisitos funcionales se describen empleando escenarios y es en este nivel donde todos los usuarios pueden optar por asignar un valor que representa la importancia de este requisito desde su punto de vista, v_{ij}.

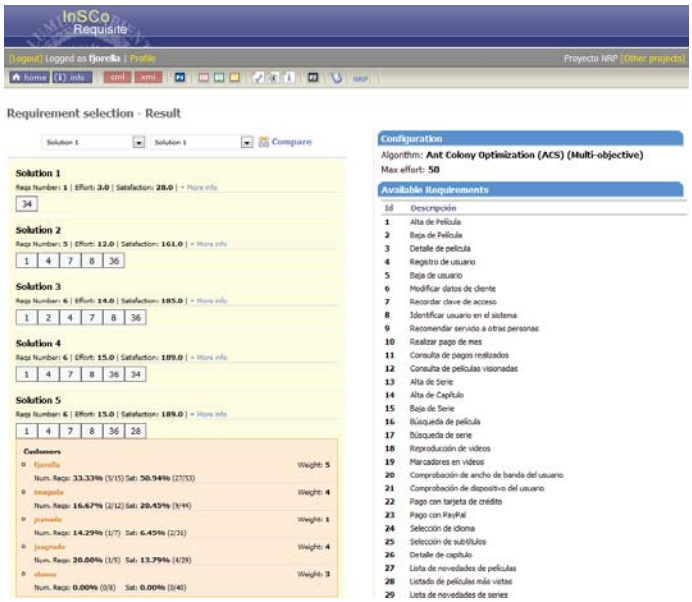


Figura 3. Interfaz de InSCo-Requisite

InSCo-Requisite trasforma la base de datos de requisitos en una representación manejable por los algoritmos de búsqueda. A partir de aquí se lanzan los algoritmos metaheurísticos, bien con los parámetros por defecto o con los que el usuario ajuste (como el número de hormigas, tamaño de la población o peso de la búsqueda local en GRASP).

Los requisitos seleccionados por el algoritmo se muestran al usuario incluyendo información sobre los recursos totales consumidos y la satisfacción. Además tal

como muestra la figura 3 los hiperenlaces permiten moverse entre la lista de requisitos seleccionados para facilitar la decisión final del experto.

6. Conclusiones

Dentro del ámbito de la ingeniería de requisitos se han definido y utilizado tres algoritmos de metaheurísticos de búsqueda, que se han incorporado dentro de una herramienta de definición y modelado de requisitos. Esta funcionalidad de InSCo-Requisite permite a los desarrolladores obtener de entre los requisitos candidatos obtenidos de los clientes la lista de aquellos que cumplen con los criterios definidos en el problema de búsqueda.

Los resultados experimentales sobre dos problemas tipo, permiten afirmar que GRAPS, los algoritmos genéticos y ACS son metaheurísticas aplicables al problema con las adaptaciones y funciones de evaluación definidas. Por otra parte, la puesta en valor de los algoritmos dentro de InSCo-Requisite, pone en práctica una arquitectura que facilita las mejoras en paralelo, tanto en las técnicas de IA como de las técnicas de modelado y gestión de requisitos.

Como trabajo futuro planteamos la mejora de las técnicas metaheurísticas, incorporando una visión multiobjetivo del problema y la mejora en la integración con el resto de etapas del desarrollo de un proyecto software así como con otras herramientas software similares.

Agradecimientos. Este trabajo se ha llevado a cabo en el marco de los proyectos del Ministerio de Educación, Cultura y Deportes TIN2010-20900-C04-02 y proyecto PIO-TEP-6174 de la Consejería de Economía, Inovación y Ciencia de la Junta of Andalucía.

Referencias

1. Chaos report: the state of the software industry. Tech. rep., Standish Group International (1994)
2. IEEE recommended practice for software requirements specifications. IEEE Std 830-1998 p. i (1998)
3. Abran, A., Moore, J., Bourque, P., Dupuis, R., Tripp, L.: Guide to the Software Engineering Body of Knowledge. IEEE Computer Society, Los Alamitos (2004)
4. van den Akker, M., Brinkkemper, S., Diepen, G., Versendaal, J.: Software product release planning through optimization and what-if analysis. Information and Software Technology 50(1-2), 101111 (2008)
5. Bagnall, A.J., Rayward-Smith, V.J., Whittley, I.: The next release problem. Information & Software Technology 43(14), 883–890 (2001)
6. Baker, P., Harman, M., Steinhöfel, K., Skaliotis, A.: Search based approaches to component selection and prioritization for the next release problem. In: ICSM. pp. 176–185. IEEE Computer Society (2006)
7. Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Jeff, S.: The agile manifiesto. Tech. rep., Agile Alliance (2001)

8. Berander, P., Svahnberg, M.: Evaluating two ways of calculating priorities in requirements hierarchies - an experiment on hierarchical cumulative voting. Journal of Systems and Software 82, 836–850 (May 2009)
9. Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B., och Dag, J.N.: An industrial survey of requirements interdependencies in software product release plannin. In: RE. pp. 84–93. IEEE Computer Society (2001)
10. Cheng, B.H.C., Atlee, J.M.: Research directions in requirements engineering. In: Briand, L.C. Wolf, A. (ed.) FOSE. pp. 285–303 (2007)
11. Davis, A.M.: The art of requirements triage. IEEE Computer 36(3), 42–49 (2003)
12. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 26(1), 29–41 (feb 1996)
13. Durillo, J.J., Zhang, Y., Alba, E., Harman, M., Nebro, A.J.: A study of the bi-objective next release problem. Empirical Software Engineering 16(1), 29–60 (2011)
14. Feo, T., Resende, M.: Greedy randomized adaptive search procedures. Journal of Global Optimization 6, 109–133 (1995)
15. Finkelstein, A., Harman, M., Mansouri, S.A., Ren, J., Zhang, Y.: A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making. Requir. Eng. 14(4), 231–245 (2009)
16. Glass, R.L.: Facts and Fallacies of Software Engineering. Addison Wesley (2002)
17. Goldberg, D.E.: Genetic Algorithms in Search Optimization and Machine learning. Addison Wesley (1989)
18. Greer, D., Ruhe, G.: Software release planning: an evolutionary and iterative approach. Information & Software Technology 46(4), 243–253 (2004)
19. Harman, M.: The current state and future of search based software engineering. In: 2007 Future of Software Engineering. pp. 342–357. FOSE '07, IEEE Computer Society, Washington, DC, USA (2007)
20. Johnson, J.: CHAOS chronicles v3.0. Tech. rep. (2003), `http://standishgroup.com/chaos/toc.php`
21. Orellana, F.J., Cañadas, J., del Águila, I.M., Túnez, S.: Insco requisite - a web-based rm-tool to support hybrid software development. In: Cordeiro, J., Filipe, J. (eds.) ICEIS 2008 (3-1). pp. 326–329 (2008)
22. Penta, M.D., Poulding, S. (eds.): Search Based Software Engineering, 2009 1st International Symposium on. IEEE (may 2009)
23. del Sagrado, J., del Águila, I.M., Orellana, F.J.: Requirements interaction in the next release problem. In: Krasnogor, N., Lanzi, P.L. (eds.) GECCO (Companion). pp. 241–242. ACM (2011)
24. del Sagrado, J., del Águila, I., Orellana, F.: Ant colony optimization for the next release problem: A comparative study. In: Search Based Software Engineering (SSBSE), 2010 Second International Symposium on. pp. 67 –76 (sept 2010)
25. del Sagrado, J., del Águila, I.M., Orellana, F.J.: Architecture for the use of synergies between knowledge engineering and requirements engineering. Lecture Notes in Computer Science 7023, 213–222 (2011)
26. Schwaber, K., Beedle, M.: Agile Software Development with Scrum. Prentice Hall, illustrated edition edn. (Oct 2001)
27. Zhang, Y., Harman, M.: Search based optimization of requirements interaction management. In: Search Based Software Engineering (SSBSE), 2010 Second International Symposium on. pp. 47 –56 (sept 2010)
28. Zhang, Y., Harman, M., Mansouri, S.A.: The multi-objective next release problem. In: Lipson, H. (ed.) GECCO. pp. 1129–1137. ACM (2007)

A. Ruíz, L. Iribarne (Eds.): Actas de las "*XVII Jornadas de Ingeniería del Software y Bases de Datos (JISBD'2012)*", Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

Réplica de un experimento que estudia las relaciones extroversión-calidad y extroversión-satisfacción en equipos de desarrollo de software

José A. Cruz-Lemus[1], Marcela Genero[1], Silvia T. Acuña[2] y Marta N. Gómez[3]

[1]Departamento de Tecnologías y Sistemas de Información,
Universidad de Castilla-La Mancha, España
{JoseAntonio.Cruz, Marcela.Genero}@uclm.es
[2]Escuela Politécnica Superior, Universidad Autónoma de Madrid, España
Silvia.Acunna@uam.es
[3]Escuela Politécnica Superior, Universidad San Pablo-CEU, España
mgomez.eps@ceu.es

Resumen. A la hora de formar un equipo de desarrollo se suelen tener en cuenta factores tales como el conocimiento y la pericia de los distintos miembros que formarán parte del equipo. Existe una tendencia, defendida por los sociólogos, que recomienda que también se tengan en cuenta factores relativos a la personalidad de los miembros del equipo, entre ellos, la extroversión de los mismos. En un estudio anterior se llevó a cabo un experimento controlado para estudiar la relación entre la extroversión de los miembros del equipo de trabajo y la calidad de los productos software obtenidos y la satisfacción percibida durante el proceso de desarrollo. En dicho estudio se concluyó que equilibrar la presencia de miembros extrovertidos y no extrovertidos en un equipo de trabajo lleva a conseguir productos de mejor calidad y, especialmente, a que la satisfacción percibida por los miembros del equipo sea muy superior que cuando los equipos sólo cuentan con miembros de carácter únicamente extrovertido o no extrovertido. Este trabajo presenta una réplica del estudio original y los resultados obtenidos confirman los resultados del experimento original para la relación positiva y directa entre los equipos con un número equilibrado de integrantes extrovertidos y no extrovertidos y la calidad de las especificaciones de requisitos software desarrolladas. Al mismo tiempo, la percepción de la satisfacción de los integrantes de los equipos sigue siendo positiva en la mayor parte de los casos.

Palabras clave: Factores de personalidad, extroversión, calidad del software, satisfacción en equipos de trabajo, especificación de requisitos software, bases de datos, réplica.

1 Introducción

A la hora de formar un equipo de desarrollo se suelen tener en cuenta factores como el conocimiento y la pericia de los distintos miembros que formarán parte del equipo.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 299-312, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

De hecho, varios estudios [1-3] han encontrado que la capacidad de los desarrolladores es uno de los factores más decisivos en el papel que juegan en un equipo, pero puede no ser el único.

Los ingenieros de software deben trabajar juntos en el desarrollo de software como un equipo, realizando, de manera coordinada, tareas independientes con relaciones complejas. De esta manera, los equipos de trabajo deben planificar los proyectos, controlar los progresos realizados y coordinar su esfuerzo, pero, a la vez, han de acordar los objetivos, tener un método de trabajo común y comunicarse regularmente [4]. Es lógico pensar, pues, que un clima de trabajo en equipo satisfactorio es esencial para llevar a cabo todas estas actividades.

Es bien conocido que la calidad del software producido por equipos de desarrollo y la satisfacción de los miembros que forman parte de estos equipos se están convirtiendo en aspectos más y más interesantes para las compañías de desarrollo de software debido a que éstas se basan cada vez más en la gestión coordinada de proyectos, reduciendo las jerarquías en el trabajo y basando su trabajo en equipos [5-7].

Todos estos antecedentes motivaron la realización de una investigación [8] sobre cómo diversos factores de personalidad (estabilidad emocional, extroversión, amabilidad, apertura al cambio y responsabilidad). En este estudio cuasi-experimental se concluyó que la extroversión media estaba directamente relacionada con la calidad del software producido y con la satisfacción percibida por los miembros del equipo de trabajo. Conviene precisar que consideramos la extroversión como un rasgo del carácter de las personas que hace que vean a los demás de una manera confiada y entusiasta. Las personas extrovertidas son, además, sociables, asertivas y comunicativas [9].

Posteriormente, y como evolución natural de esta investigación cuasi-experimental, se diseñó y llevó a cabo un experimento controlado [10] para evaluar la validez de las conclusiones inicialmente alcanzadas. Los resultados de este experimento controlado indicaban que tanto la calidad de los productos software desarrollados como la satisfacción percibida por los miembros de un equipo de desarrollo era mayor si se equilibraba la presencia de miembros extrovertidos y no extrovertidos en el equipo. El experimento se llevó a cabo en un contexto académico, en concreto, en el desarrollo de una especificación de requisitos para el desarrollo de una base de datos.

La replicación de trabajos empíricos es fundamental para conseguir un mayor poder confirmatorio en los resultados obtenidos [11], es por ello que en este trabajo se presenta una réplica de este experimento controlado, realizada en el mismo contexto, pero con una diferencia temporal de un curso académico. Con ella se pretende confirmar los hallazgos del experimento controlado original, con el fin de obtener conclusiones robustas sobre las relaciones extroversión-calidad y extroversión-satisfacción en equipos de desarrollo de software.

El resto del trabajo se organiza de la siguiente manera: la sección 2 presenta una serie de trabajos relacionados, con el fin de contextualizar la investigación que se ha desarrollado. En la sección 3 se describe el experimento original que se ha replicado. Todos los detalles relativos a la réplica constituyen la sección 4. La sección 5 se utiliza para discutir los resultados obtenidos. La sección 6 señala las principales amenazas a la validez experimental. Para finalizar, la sección 7 resume las principales conclusiones obtenidas y el trabajo futuro a realizar.

2 Trabajo relacionado

El software es desarrollado por personas, utilizado por personas y ayuda a realizar el trabajo a las personas. Esto indica la importancia que tiene el componente humano en el desarrollo del software [12]. Se han realizado investigaciones que tienen en cuenta este aspecto e incorporan a las personas en el proceso de software [13-18]. Estos trabajos analizan individualmente las personas y establecen sus relaciones con las actividades realizadas dentro del proyecto.

Sin embargo, la ingeniería de software es una actividad esencialmente de equipo. En el equipo de desarrollo de software, los miembros desempeñan diferentes roles en el proyecto para realizar las distintas actividades que lo componen. De esta manera, el software desarrollado responderá a sus requisitos en función de lo que hagan o dejen de hacer los equipos y sus miembros durante el desarrollo de software. Además, las actividades que forman este proceso están interrelacionadas y exigen que las personas implicadas se coordinen y comuniquen adecuadamente para que con su trabajo se logre el éxito del proyecto. A los ingenieros de software no sólo se les exige unos conocimientos técnicos sino que deben ser capaces de trabajar en equipo para lograr la calidad óptima en el software desarrollado. Por tanto, es importante la configuración del equipo, siendo la personalidad de los miembros del mismo un aspecto relevante a considerar. Con esta idea, se han llevado a cabo investigaciones en Ingeniería del Software que tratan de determinar la influencia de la personalidad en el desarrollo de software.

Así, algunos investigadores han examinado el efecto que tienen la personalidad de los integrantes del equipo, las características de la estructura del equipo y los modos de comunicación sobre la productividad del equipo en el campo del desarrollo de sistemas de información [3, 19-20].

Hay estudios que utilizan un test estándar, como el Myers-Briggs Type Indicator (MBTI) [21-26], para determinar las directrices para el éxito del equipo según los tipos de personalidad de los ingenieros del software e identificar una serie de rasgos y características que puedan ayudar en la formación de un equipo eficaz. El estudio de [27] determina la conexión entre las habilidades, los rasgos de personalidad y el rendimiento del equipo. Este estudio se efectuó sobre equipos de profesionales muy consolidados y el factor clave examinado fue si las tareas a realizar eran o no rutinarias.

En [28] se propone el uso de un cuestionario para llevar a cabo un estudio experimental de tipo descriptivo en el que se analiza la relación entre la personalidad y capacidades de los integrantes del equipo con respecto a la eficacia del mismo. Se comprueba que el uso de este cuestionario ayuda a los equipos a mejorar su eficacia y a resolver conflictos en el equipo. El instrumento mejora el conocimiento de las capacidades y rasgos de personalidad de los integrantes del equipo, incluyendo fortalezas y debilidades que resultan útiles en situaciones críticas para el equipo.

La investigación llevada a cabo por [29] presenta un estudio experimental de tipo descriptivo y correlacional en el que se analiza el impacto de la personalidad en relación al éxito del proyecto. Se comprueba que la personalidad de los miembros del equipo tiene un impacto significativo sobre el éxito del proyecto, mientras que no ocurre lo mismo con la diversidad de personalidades dentro del equipo.

La investigación desarrollada por [30] fue un experimento controlado de tipo descriptivo y correlacional, en el que se analiza la relación de la personalidad respecto a la eficacia del equipo (programación por pares), calidad de desarrollo y satisfacción del equipo. Se hace una comparación entre pares homogéneos y heterogéneos en términos de la eficacia del par. Los resultados muestran que hay diferencias importantes entre los grupos heterogéneos y homogéneos. Los pares con temperamentos y personalidades heterogéneas presentan mejor comunicación, eficacia y viabilidad de la colaboración. En [31] se realiza un estudio empírico que investiga sobre: la naturaleza y los efectos de la personalidad en la colaboración de la programación por pares. Se trata de un estudio experimental utilizando un análisis de árbol de decisión. Una de sus conclusiones fue que la heterogeneidad de personalidades aumenta la cantidad de comunicación e intensidad de la colaboración.

Algunos estudios experimentales correlacionales [32-33] analizan la influencia que tienen algunos factores de personalidad como la responsabilidad o la estabilidad emocional sobre el rendimiento de los desarrolladores de software que practican la programación por pares. En el primero, los resultados indicaron que la responsabilidad no afectaba significativamente al rendimiento, lo cual podía deberse a la corta duración de las tareas realizadas a lo largo de todo el experimento. Sin embargo, los resultados revelaron que otro factor de personalidad, la apertura al cambio, presentaba una correlación directa positiva con el rendimiento. El segundo experimento no encontró ninguna relación entre la estabilidad emocional y el rendimiento.

El trabajo [34] presenta una revisión sistemática de la literatura acerca de la personalidad en ingeniería de software donde se consideran algunas de las investigaciones anteriores. Los resultados indican que la mayoría de los estudios analizados son investigación empírica sobre la influencia que tiene la personalidad, tanto en programación por pares como sobre la eficacia de los equipos. También señala que hace falta realizar replicaciones sobre los estudios empíricos y probar los modelos propuestos en los estudios teóricos a través de estudios empíricos. Una conclusión que se extrae es la necesidad de llevar a cabo replicaciones de los estudios empíricos para lograr la consolidación del conocimiento obtenido que pueda servir para nuevas investigaciones y la influencia sobre la práctica de la ingeniería del software.

Siguiendo la línea de investigación de los factores de personalidad y su impacto en el desarrollo de software, [8] diseña un cuasi-experimento correlacional con el que analizan la relación entre personalidad, procesos de equipo, características de la tarea, calidad del software y satisfacción del equipo de desarrollo. Los equipos aplican una adaptación de la metodología ágil (eXtreme Programming, XP) para desarrollar un producto software. Se encontró que los equipos con mayor satisfacción eran precisamente aquellos cuyos miembros presentaban puntuaciones más altas para los factores de personalidad amabilidad y la responsabilidad. Los niveles de satisfacción son también más altos cuando los miembros del equipo pueden decidir cómo organizarse y desarrollar su trabajo. Por otra parte, el nivel de satisfacción y cohesión disminuye cuanto mayor es el grado de conflictividad entre los miembros del equipo. Por último, los equipos muestran una correlación positiva directa entre la media del factor de personalidad extroversión y la calidad del producto software. Tal y como se mencionó anteriormente, este cuasi-experimento permite estudiar correlaciones pero no relacio-

nes causales. Partiendo de esta investigación, se diseñó y realizó un experimento controlado en el ámbito académico [10] para corroborar algunos de los resultados obtenidos. Las conclusiones fueron que se incrementa tanto la calidad de los productos software desarrollados como la satisfacción del equipo de desarrollo cuando su configuración presenta un equilibrio entre miembros extrovertidos y no extrovertidos.

3 Experimento original

El experimento original se llevó a cabo durante el curso académico 2010/2011 con los 76 alumnos de la asignatura de Bases de Datos de las distintas titulaciones de la Escuela Superior de Informática de Ciudad Real (Universidad de Castilla-La Mancha).

Se formaron 19 equipos de trabajo que, como parte de la evaluación de la asignatura, debían desarrollar un sistema de bases de datos completo. Cada uno de los sujetos realizó previamente un test de personalidad (basado en [9]) que lo caracterizaba como extrovertido o no extrovertido. Después los sujetos se distribuyeron en 3 tipos de equipos (ver Tabla 1).

Tabla 1. Tipos de equipos según la extroversión de sus componentes (experimento)

Tipo	Composición	Equipos
EXT	4 sujetos extrovertidos	6
MIX	2 sujetos extrovertidos y 2 no extrovertidos	7
NO-EXT	4 sujetos no extrovertidos	6

El trabajo a realizar por cada uno de los equipos consistía en la especificación de los requisitos del sistema (ERS) en lenguaje natural. A partir de esta lista de requisitos, los profesores responsables de la asignatura realizaban una corrección de los mismos y conseguían una valoración subjetiva de la calidad de la ERS que se calculaba dividiendo el número de defectos encontrados (según la plantilla propuesta en [35]) por el número de entidades del diagrama E/R al que daba lugar la ERS.

Además, cada uno de los miembros del equipo de trabajo debía rellenar un formulario (ver ANEXO 1) en el que valoraba subjetivamente una serie de preguntas relativas a la satisfacción percibida durante el desarrollo del trabajo.

Los resultados obtenidos indicaban que los equipos MIX habían producido los entregables de mayor calidad y que, además, en estos mismos equipos, la satisfacción percibida por los miembros del equipo había sido superior al resto de tipos de equipo.

Si bien estos resultados no mostraban relaciones estadísticamente significativas entre extroversión-calidad y extroversión-satisfacción, sí que presentaban indicios que apuntaban a los equipos MIX como los más interesantes para mejorar estas relaciones.

Las réplicas de estudios empíricos son fundamentales para conseguir un mayor poder confirmatorio de los resultados [11], por lo que se decidió realizar la réplica que se describirá en la siguiente sección, con el fin de confirmar y fortalecer los resultados preliminares obtenidos en el experimento.

4 Réplica del experimento

En las siguientes sub-secciones se proporcionan todos los detalles relativos al diseño de la réplica según las líneas guía propuestas en [36].

4.1 Motivación de la replica

Como se comentó en la sección anterior, los resultados obtenidos en el experimento original mostraban indicios favorables a la presencia equilibrada de miembros extrovertidos y no extrovertidos (equipos MIX) en equipos de desarrollo, si bien dichos resultados no eran estadísticamente significativos.

Así pues, el objetivo principal de esta réplica consiste en confirmar si, como ocurrió en el experimento original, los equipos MIX producen desarrollos de mayor calidad y permiten que los miembros del equipo consigan una mayor satisfacción percibida durante el proceso de trabajo.

4.2 Contexto y selección de sujetos

Un total de 78 alumnos han participado como sujetos experimentales, todos ellos matriculados en la asignatura de Bases de Datos de la Ingeniería en Informática y las Ingenierías Técnicas de Informática de Gestión y de Sistemas, impartidas en la Escuela Superior de Informática de Ciudad Real (Universidad de Castilla-La Mancha).

De nuevo, como en el experimento original, los sujetos tenían que desarrollar y entregar la ERS en lenguaje natural de un sistema de base de datos elegido por ellos, teniendo como restricción que el diagrama E/R asociado tuviera entre 15 y 20 entidades.

Tras realizar el test de personalidad, se categorizó a la mitad de los sujetos como extrovertidos y a la otra mitad como no extrovertidos y se distribuyeron en 20 equipos de trabajo (18 de 4 personas y 2 de 3 personas), según se muestra en la Tabla 2.

Tabla 2. Tipos de equipos según la extroversión de sus componentes (réplica)

Tipo	Composición	Equipos
EXT	Todoos los sujetos extrovertidos	6
MIX	Número de sujetos extrovertidos y no extrovertidos balanceado	8
NO-EXT	Todos los sujetos no extrovertidos	6

El resto de características de la réplica son similares a las del experimento original y se irán comentando en las siguientes sub-secciones.

4.3 Diseño de la replica

El diseño experimental elegido es, como en el experimento original, un diseño inter-sujetos con un único factor (extroversión del equipo) con los tres posibles tratamientos que ya se han comentado con anterioridad (EXT, MIX y NO-EXT).

4.4 Variables seleccionadas

La única variable independiente del estudio es la extroversión del equipo (EE), ya que el objetivo de esta investigación es estudiar si esta variable afecta tanto a la calidad de la especificación de requisitos software como a la satisfacción percibida por los miembros de un equipo de desarrollo. Se trata de una variable con escala nominal que, como ya se ha comentado, puede tomar 3 posibles valores (ver Tabla 2).

En cuanto a las variables dependientes, en este estudio se cuenta con 2:

- *Calidad ERS*: esta variable se calcula dividiendo en número de defectos encontrados en la ERS por el número de entidades del diagrama E/R asociado a la misma. Para calcular el número de defectos se utilizó una adaptación de la lista de comprobación propuesta en [35] (ver Tabla 3). Se trata de una variable en escala de ratio.

Tabla 3. Tipos de defectos de la ERS

Tipo de defecto	Descripción
Omisión	Información que debería haber sido incluida pero que se he ha omitido
Inconsistencia	Información contradictoria que genera una ERS inconsistente
Ambigüedad	Información poco clara o imprecisa que puede malinterpretarse
Redundancia	Información redundante o innecesaria que no se utilizará
Miscelánea	Defectos asociados al uso incorrecto del lenguaje natural

- *Satisfacción*: esta variable captura la percepción de los sujetos al trabajar en equipos. Se obtiene a través de un cuestionario, basado en [37] (ver ANEXO 1) con afirmaciones que deben ser valoradas en una escala Likert de 5 puntos. El coeficiente alfa de Cronbach, como indicador de fiabilidad del instrumento, es de 0,90. En esta ocasión, se trata de una variable en escala ordinal.

4.5 Formulación de hipótesis

Se formularon dos hipótesis experimentales según se muestra a continuación:

- H_{10}: No hay diferencia en la calidad de la ERS en equipos de trabajo con diferente tipo de extroversión. $H_{11}=\neg H_{10}$
- H_{20}: No hay diferencia en la satisfacción en equipos de trabajo con diferente tipo de extroversión. $H_{11}=\neg H_{10}$

A través del análisis estadístico que se presentará más adelante se comprobará si se pueden rechazar las hipótesis nulas que se han planteado en la investigación a través de los datos recolectados.

4.6 Preparación, ejecución y análisis de datos

El trabajo a desarrollar por parte de los sujetos era parte de la evaluación de la asignatura de bases de datos y constituía una parte imprescindible para poder aprobarla, por lo que no se consideró necesario ningún tipo de motivación adicional para que la llevaran a cabo correctamente. Antes de la realización de la ERS, todos habían recibido formación específica sobre el tema en cuestión, habían hecho y corregido ejercicios sobre modelado conceptual utilizando el modelo E/R e, incluso, habían realizado un examen parcial sobre el tema, por lo que se consideró que no era necesario proporcionarles ningún otro tipo de formación original. En cualquier caso, durante la realización del trabajo podían asistir a las tutorías de sus profesores responsables en cualquier momento para resolver cuantas dudas y cuestiones les fueran surgiendo.

El envío de los trabajos y de los tests se satisfacción se realizó mediante la plataforma de campus virtual de la Universidad de Castilla-La Mancha, basada en moodle y los plazos de entrega se publicaron con varios meses de antelación.

Tras la recolección de todo el material se utilizó el programa estadístico SPSS (v.19) para realizar todos los cálculos relativos al análisis de los datos. En concreto, para la variable *calidad ERS* se llevó a cabo un análisis descriptivo de los estadísticos y un ANOVA para la comprobación de la hipótesis experimental H_{10}. Además, para analizar la variable *satisfacción* se realizó un análisis de frecuencias acumuladas para cada una de las preguntas del cuestionario.

Todos resultados obtenidos, así como la interpretación de los mismos, se detallan en la siguiente sección.

5 Resultados

En esta sección se comentan los resultados obtenidos tras la realización del análisis estadístico. Para facilitar la lectura, se ha estructurado en dos sub-secciones: la primera explora la relación entre la extroversión y la calidad de la ERS y la segunda se centra en la relación entre la extroversión y la satisfacción percibida por los miembros del equipo de desarrollo.

5.1 Relación extroversión-calidad ERS

Los estadísticos descriptivos asociados a la variable *calidad ERS* se muestran a continuación de forma numérica (Tabla 4).

Como puede apreciarse, los mejores resultados se obtienen siempre para los grupos MIX, ya que la proporción de errores por entidad es la mínima. Estos resultados se encuentran en total concordancia con los obtenidos en el experimento original, donde se obtuvieron medias muy similares (EXT: 1,4103; MIX: 1,0252; NO-EXT: 1,2461).

Tabla 4. Estadísticos descriptivos para la variable *calidad ERS*

Tipo	n	Media	Desv. Tip.
EXT	6	1,6410	1,591
MIX	8	**0,7905**	0,664
NO-EXT	6	1,0437	1,056

Con el fin de contrastar la hipótesis experimental H_{10} se llevó a cabo un ANOVA. El nivel de significación de la variable *calidad SRS* es de 0,385 y la potencia observada del 19,7%. Estos valores no nos permiten rechazar la hipótesis H_{10}, si bien están completamente en concordancia con los valores obtenidos en el experimento original, lo que parece indicar una tendencia en la que los equipos MIX obtienen entregables de mayor calidad que el resto de tipos de equipo.

5.2 Relación extroversión-satisfacción

A continuación, en las Tablas 5, 6 y 7, se muestran los resultados obtenidos en el análisis de frecuencias acumuladas para cada una de las preguntas del test de satisfacción personal que cada sujeto debía entregar tras la realización del trabajo. En tabla se muestra el número de respuestas para cada opción (columna FAbs.), el porcentaje relativo (columna %), y la frecuencia acumulada expresada en porcentaje (columna FAcum.), además del número de respuestas procesadas en cada tipo de equipo. Hay que puntualizar que no todos los sujetos respondieron el test.

Los valores de la columna Respuesta corresponden a: 5-Totalmente de acuerdo; 4-De acuerdo; 3-Neutral; 2-En desacuerdo; 1-Totalmente en desacuerdo.

Tabla 5. *Satisfacción:* Tabla de frecuencias para la primera pregunta (ver Anexo 1)

	EXT (n=19)			MIX (n=23)			NO-EXT (n=16)		
Respuesta	FAbs	%	FAcum	FAbs	%	FAcum	FAbs	%	FAcum
5	9	47,4	47,4	7	30,4	30,4	2	12,5	12,5
4	7	36,8	**84,2**	9	39,2	**69,6**	10	62,5	**75,0**
3	2	10,5	94,7	4	17,4	87,0	1	6,3	81,3
2	1	5,3	100	2	8,7	95,7	2	12,5	93,8
1	0	0	100	1	4,3	100	1	6,2	100

Tabla 6. *Satisfacción:* Tabla de frecuencias para la segunda pregunta (ver Anexo 1)

	EXT (n=19)			MIX (n=23)			NO-EXT (n=16)		
Respuesta	FAbs	%	FAcum	FAbs	%	FAcum	FAbs	%	FAcum
5	9	47,4	47,4	2	8,7	8,7	0	0	0
4	6	31,5	**79,9**	12	52,2	**60,9**	10	62,5	**62,5**
3	4	21,1	100	7	30,4	91,3	4	25,0	87,5
2	0	0	100	2	8,7	100	2	12,5	100
1	0	0	100	0	0	100	0	0	100

Tabla 7. *Satisfacción:* Tabla de frecuencias para la tercera pregunta (ver Anexo 1)

	EXT (n=19)			MIX (n=23)			NO-EXT (n=16)		
Respuesta	FAbs	%	FAcum	FAbs	%	FAcum	FAbs	%	FAcum
5	11	57,9	**57,9**	5	21,8	21,8	2	12,5	12,5
4	6	31,6	89,5	11	47,8	**69,6**	11	68,8	**81,3**
3	2	10,5	100	5	21,8	91,4	2	12,5	93,8
2	0	0	100	1	4,3	95,7	1	6,2	100
1	0	0	100	1	4,3	100	0	0	100

Con el fin de aumentar la legibilidad de las tablas, se han destacado las celdas en las que se encuentra la mediana de cada una de las series de valores, es decir, aquellos valores en los que se alcanza un 50% por ciento de los valores acumulados.

Si se observan las valoraciones positivas de los sujetos, aquellas en las que está de acuerdo o totalmente de acuerdo con la afirmación a la que respondía, puede apreciarse como en todos los casos los equipos EXT obtienen siempre una valoración superior que en el resto de los grupos. En el experimento original se daba esta misma situación pero con los equipos MIX. Una posible explicación para estos datos radica en que la composición de los equipos EXT, formados únicamente por miembros extrovertidos, haga que éstos tengan una visión más positiva y entusiasta de la realidad, a pesar de que como se vio en la sección anterior (Tabla 4) los resultados que obtienen son los de peor calidad.

Aún así, entre el 60 y el 70% de los componentes de los equipos MIX obtienen una satisfacción positiva.

5.3 Discusión de los resultados

En lo relativo a la variable *calidad ERS*, y como ya ocurrió en el experimento original, se ha encontrado que los equipos MIX, en los que se equilibra el número de miembros extrovertidos y no extrovertidos, obtienen los mejores resultados y producen las ERS de mayor calidad.

Respecto de la variable *satisfacción*, en este caso no se confirman los hallazgos del experimento original, puesto que en esta ocasión son los equipos EXT (con todos sus integrantes extrovertidos) los que han percibido una satisfacción mayor durante la realización del trabajo. Una posible explicación a esta diferencia en los resultados es que, en la réplica, los sujetos tuvieron menos tiempo para realizar el entregable por lo que las relaciones interpersonales probablemente alcanzaron un nivel de madurez inferior que en el experimento original. En esta situación, cabe pensar que los equipos únicamente con miembros extrovertidos pudieran haber percibido una mejor satisfacción durante el proceso de desarrollo.

En cualquier caso, se sigue observando cómo los equipos MIX, en los que se equilibra el número de miembros extrovertidos y no extrovertidos obtienen los resultados de mayor calidad al tiempo que la satisfacción percibida por los miembros de estos equipos es positiva entre un 60 y un 70% de los casos.

6 Amenazas a la validez

A continuación se analizan diversos factores que pueden haber amenazado la validez de la réplica y qué medidas se han aplicado para mitigarlos o eliminarlos.

- Validez interna: puede haberse visto afectada debido a que los sujetos elegían el dominio sobre el cual querían desarrollar su ERS libremente. En cualquier caso, el tamaño de todos los trabajos era similar al ser necesario que el diagrama E/R asociado a la ERS tuviera entre 15 y 20 entidades, por lo que la complejidad no debería variar sustancialmente entre los distintos trabajos.
- Validez externa: los sujetos estaban cursando la asignatura de Bases de Datos mientras realizaban su trabajo, por lo que podrían considerarse como desarrolladores novatos. Aún así, se habían realizado suficientes ejercicios sobre todo el marco teórico necesario para la realización del mismo.
- Validez de la conclusión: los tests estadísticos utilizados están comúnmente aceptados para el tipo de diseño experimental elegido [38-39], por lo que la validez de la conclusión no debería verse afectada.
- Validez del constructo. Este tipo de validez tampoco se ha visto afectado, ya que se han utilizado los materiales estándar recomendados por la literatura para realizar la recolección de datos [9, 37], así como la corrección de los entregables [35].

7 Conclusiones y trabajo futuro

Este trabajo presenta una réplica de un experimento realizada con el fin de analizar si la extroversión de los miembros de un equipo de trabajo afecta, por un lado, a la calidad de las especificaciones de requisitos software desarrollados y, por otro, a la satisfacción percibida por los miembros del equipo durante la realización del trabajo.

En el marco del desarrollo de la asignatura de Bases de Datos de distintas titulaciones se agrupó a los alumnos en equipos de trabajo de manera que en unos equipos todos los miembros eran extrovertidos, en otro tipo de equipo todos eran no extrovertidos y, en el tercer tipo, se equilibró la presencia de unos y otros.

Todos los equipos desarrollaron una especificación de requisitos software de un sistema de complejidad similar y, después, rellenaron un cuestionario en el que dejaban constancia de la satisfacción que habían percibido durante la realización del trabajo.

Los resultados obtenidos indican que equilibrar el número de integrantes extrovertidos y no extrovertidos en un equipo de trabajo hace que se desarrollen especificaciones de requisitos software de mayor calidad, a la vez que la percepción de la satisfacción de los integrantes de estos equipos sea positiva en la mayor parte de los casos.

Estos resultados están alineados con los obtenidos en el experimento original, si bien sería muy recomendable como trabajo futuro realizar una nueva réplica introduciendo una serie de cambios en el diseño experimental. Entre estos cambios cabe destacar la utilización de desarrolladores de mayor experiencia y, a ser posible,

vinculados al mundo profesional del desarrollo de software o la utilización de un único sistema software común para todos los equipos de desarrollo con el fin de evitar posibles diferencias en la complejidad de sistemas diferentes y considerando, además, el desarrollo del sistema completo, no sólo la especifiación de requisitos.

De volver a confirmar los resultados obtenidos, se podría establecer una recomendación, aplicable tanto al mundo empresarial como al académico, consistente en que a la hora de formar equipos de trabajo se tenga en cuenta la extroversión del los miembros del mismo y, en la medida de lo posible, se trate de equilibrar la presencia de miembros extrovertidos y no extrovertidos. A la luz de los resultados obtenidos en este estudio, esta distribución permitirá obtener resultados de mayor calidad en lo concerniente a especificaciones de requisitos software, mientras que el clima de trabajo y las relaciones interpersonales en el equipo se valorará positivamente en la gran mayoría de los casos.

Agradecimientos. Esta investigación se ha financiado gracias a los siguientes proyectos: MEDUSAS (CDTI-MICINN and FEDER IDI-20090557), ORIGIN (CDTI-MICINN and FEDER IDI-2010043(1-5)), PEGASO/MAGO (MICINN and FEDER, TIN2009-13718-C02-01), EECCOO (MICINN TRA2009_0074), MECCA (JCMM PII2I09-0075-8394), IMPACTUM (JCCM PEII11-0330-4414), Tecnologías para la Replicación y Síntesis de Experimentos en IS (MICINN TIN2011-23216) and Go Lite (MICINN TIN2011-24139).

Referencias

1. Boehm, B.W. Improving Software Productivity. IEEE Computer 20(9), 43-57 (1987)
2. Curtis, B., Krasner, H., Iscoe, N. A Field Study of the Software Design Process for Large Systems. Comm. ACM 31(11), 1268–1287 (1988)
3. Rasch, R.H., Tosi, H.L. Factors Affecting Software Developers Performance: An Integrated Approach. MIS Quarterly 16(3), 395-409 (1992)
4. Curtis, W., Miller, S. Hefley, W. People Capability Maturity Model (P-CMM) Version 2.0, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Maturity Module CMU/SEI-2001-MM-001 (2001)
5. Chung, W.Y., Guinan, P.J. Effects of Participative Management on the Performance of Software Development Teams. In: 1994 Computer Personal Research Conference on Reinventing I, 252-260 (1994)
6. Faraj, S., Sproull, L. Coordinating Expertise in Software Development Teams. Manag. Sci. 46(12),1554-1568 (2000)
7. Yang, H.L., Tang, J.H. Team Structure and Team Performance in IS Development: a Social Network Perspective. Inf. Manag. 41, 335-349 (2004)
8. Acuña, S.T., Gómez, M., Juristo, N. How do personality, team processes and task characteristics relate to job satisfaction and software quality? Inf. Soft. Tech. 51(3), 627-639 (2009)
9. Costa Jr., P.T., McCrae, R.R. NEO Personality Inventory-Revised. Psychological Assessment Resources, Odessa, FL (1992)

10. Acuña, S.T., Genero, M.F., Gómez, M.N., Cruz-Lemus, J.A., Juristo, N. (2012) Does team extraversion impact on software quality and team satisfaction? A controlled experiment. Emp. Soft. Eng. (submitted)
11. Brooks, A., Rooper, M., Wood, M., Daly, J., Miller, J. Replication´s Role in Software Engineering. In: Shull, F., Singer, J., Sjberg, D. (eds.) Guide to Empirical Software Engineering (Chapter 14) Springer, Heidelberg (2008)
12. DeSouza, C.R.B., Sharp, H. , Singer, J., Cheng, L., Venolia, G. Cooperative and Human Aspects of Software Engineering. IEEE Soft. 26(6) 17-19 (2009)
13. Moore, E. Personality characteristics of information systems professionals. In: Conference on SIGCPR, 140–155 (1991)
14. Turley, R., Bieman, J. Competencies of exceptional and non-exceptional software engineers. J. Syst. Soft. 28(1), 19-38 (1995)
15. Kellner, M.I., Madachy, R.J., Raffo, D.M. Software process simulation modelling: Why? What? How? J. Syst. Soft. 46, 91-105 (1999)
16. Wynekoop, J., Walz, D. Investigating traits of top performing software developers. Inf. Tech. People 13(3), 186-195 (2000)
17. Acuña, S.T., Juristo, N. Assigning people to roles in software projects. Soft.: Pract. Exp. 34(7), 675-696 (2004)
18. Pfahl, D., Laitenberger, O., Ruhe, G., Dorsch, J., Krivobokova, T. Evaluating the learning effectiveness of using simulations in software project management education: Results from a twice replicated experiment. Inf. Soft. Tech. 46, 127-147 (2004)
19. Borovits, I., Ellis, S., Yeheskel, O. Group Processes and the Development of Information Systems. Inf. Man. 19, 65-72 (1990)
20. White, K.B., A Preliminary Investigation of Information Systems Team Structures. Inf. Man. 7(6), 331-335 (1984)
21. Bostrom, R.P., Kaiser, K.M. Personality differences within systems project teams: implications for designing solving centers. In: 18th Annual ACM SIGCPR Conference, 248-285 (1981)
22. Hardiman, L.T. Personality types and software engineers. IEEE Comp. 301(10), 10-10 (1997)
23. Teague, J. Personality type, career preference and implications for computer science recruitment and teaching. In: 3rd Australasian Conference on Computer Science Education, 155–64 (1998)
24. Rutherfoord, R.H. Using personality inventories to help form teams for software engineering class projects. In: ACM SIGCSE Bulletin, 6th Annual Conference on Innovation and Technology in Computer Science Education 33(3), 73-76 (2001)
25. Capretz, L.F. Personality types in software engineering. Int. J. Hum.-Comp. Stud. 58(2), 207–214 (2003)
26. Karn J., Cowling, T. A follow up study of the effect of personality on the performance of software engineering teams. In: 2006 ACM/IEEE International Symposium on Empirical Software Engineering, 232–241 (2006)
27. White, K., Leifer, R. Information systems development success: perspectives from project team participants. MIS Quarterly 10(3), 215-23 (1986)
28. Zuser, W., Grechenig, T. Reflecting skills and personality internally as means for team performance improvement. In: 16th Conference on Software Engineering Education and Training, 234-241 (2003)
29. Peslak, A.R. The Impact of Personality on Information Technology Team Projects. In: ACM Conference on Computer Personnel Research (SIGMISCPR' 06), 273-279 (2006)

30. Sfetsos, P., Stamelos, I., Angelis, L., Deligiannis, I. An Experimental Investigation of Personality Types Impact on Pair Effectiveness in Pair Programming. Emp. Soft. Eng. 14, 187-226 (2009)
31. Walle, T., Hannay, J.E. Personality and the Nature of Collaboration in Pair Programming. In: 3rd International Symposium on Empirical Software Engineering and Measurement, 203-213 (2009)
32. Salleh, N., Mendes, E., Grundy, J.C., Burch, G.S.J. An Empirical Study of the Effects of Conscientiousness in Pair Programming Using the Five-factor Person-ality Model. In: ACM/IEEE International Conference on Software Engineering 1, 577-586 (2010)
33. Salleh, N., Mendes, E., Grundy, J.C., Burch, G.S.J. The Effects of Neuroticism on Pair Programming: An Empirical Study in the Higher Education Context. In: ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, (2010)
34. Da Silva, F.Q.B., Cruz, S., Monteiro, C., Santos, P., Rossilei, I. Personality in Software Engineering: Preliminary Findings from a Systematic Literature Review. In: 15th Conference on Evaluation & Assessment in Software Engineering (EASE 2011), 1-10, (2011)
35. Carver, J.C., Nagappan, N., Page, A. The impact of educational background on the effectiveness of requirements inspections: an empirical study. IEEE Trans.Soft.Eng. 34(6), 800–812 (2008)
36. Carver, J., Towards Reporting Guidelines for Experimental Replications: A Proposal. In: 1st International Workshop on Replication in Empirical Software Engineering Research (2010)
37. Gladstein, D.L. Groups in context: a model of task group effectiveness. Administrative Science Quarterly. 29(4), 499–517 (1984)
38. Kirk, R.E. Experimental Design: Procedures for the Behavioral Sciences. Brooks/Cole Publishing Company (1995)
39. Winer B.J., Brown D.R., Michels K.M. Statistical Principles in Experimental Design. McGraw-Hill. (1991)

Anexo 1: Formulario para percepción de la satisfacción

A - Totalmente de acuerdo	B – De acuerdo	C – Neutral	D - Desacuerdo	E - Totalmente en desacuerdo

1 – Estoy muy satisfecho con el hecho de haber trabajado en este equipo	A	B	C	D	E
2 – Estoy encantado con la forma en que mis compañeros y yo trabajamos juntos	A	B	C	D	E
3 – Estoy satisfecho con mis compañeros actuales	A	B	C	D	E

Sesión Temática 4

Calidad, Pruebas y Requisitos

Coordinadores: *Dr. Xavier Franch y Dr. Claudio de la Riva*

Sesión Temática 4: Calidad, Pruebas y Requisitos
Coordinadores: Dr. Xavier Franch y Dr. Claudio de la Riva

Federico Leonardo Toledo, Beatriz Pérez Lamancha and Macario Polo. *Enfoque dirigido por modelos para probar Sistemas de Información con Bases de Datos*. (Regular)

Raquel Blanco, Javier Tuya and Ruben V. Seco. *Evaluación de la cobertura en la interacción usuario-base de datos utilizando un enfoque de caja negra*. (Regular)

Juan Jose Dominguez-Jimenez, et al. *Evolutionary Mutation Testing*. (Relevante)

Carmen R. Cutilla, Julian A. García-García and Javier J. Gutiérrez. *Hacia una propuesta de priorización de casos de pruebas a partir de NDT*. (Emergente)

Silvio Cacace and Tanja Vos. *Model-Based Testing in Early Software Development Phases*. (Herramienta)

Antonia Estero-Botaro, et al. *Operadores de Mutación de Cobertura para WS-BPEL 2.0*. (Regular)

Lorena Gutiérrez-Madroñal, Juan José Domínguez-Jiménez and Inmaculada Medina-Bulo. *Prueba de mutaciones sobre consultas de procesamiento de eventos en aplicaciones en tiempo real*. (Regular)

Marcos Palacios, José García-Fanjul and Javier Tuya. *Testing in Service Oriented Architectures with dynamic binding: A mapping study*. (Relevante)

Sergio Segura, et al. *Automated Metamorphic Testing on the Analysis of Feature Models*. (Relevante)

Ana Belén Sánchez and Sergio Segura. *Automated testing on the analysis of variability-intensive artifacts: An exploratory study with SAT Solvers*. (Emergente)

César J. Pardo Calvache, et al. *PrMO: An Ontology of Process-reference Models*. (Regular)

Albert Tort, Antoni Olivé and Maria-Ribera Sancho. *An Approach to Test-Driven Development of Conceptual Schemas*. (Relevante)

Victor M. R. Penichet, et al. *Requirement-based Approach for Groupware Environments Design*. (Relevante)

Emma Blanco-Muñoz, et al. *GAmeraHOM: una herramienta de generación de mutantes de orden superior para WS-BPEL*. (Herramienta)

Antonio García Domínguez, et al. *MuBPEL: una Herramienta de Mutación Firme para WS-BPEL 2.0*. (Herramienta)

Federico Leonardo Toledo, Macario Polo and Beatriz Pérez Lamancha. *Tutorial de Pruebas de Rendimiento*. (Tutorial)

A. Ruíz, L. Iribarne (Eds.): Actas de las "*XVII Jornadas de Ingeniería del Software y Bases de Datos (JISBD'2012)*", Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

Enfoque dirigido por modelos para probar Sistemas de Información con Bases de Datos

Federico Toledo Rodríguez[1] , Beatriz Pérez Lamancha[2], Macario Polo Usaola[3],

[1] Abstracta, Montevideo, Uruguay.
ftoledo@abstracta.com.uy
[2]Centro de Ensayos de Software, Universidad de la República, Montevideo, Uruguay
bperez@fing.edu.uy
[3]Universidad de Castilla-La Mancha, Ciudad Real, España
macario.polo@uclm.es

Abstract. La base de datos es un componente esencial de los sistemas de información. En efecto, para una base de datos dada, una organización tendrá, probablemente, múltiples aplicaciones que la gestionen, de acuerdo a los diferentes tipos de usuarios, plataformas, dispositivos, etcétera. Aunque la propia base de datos ya incorporará su propio conjunto de restricciones, la lógica de los programas asociados a ella debe también contemplarlas y manejarlas correctamente. En este artículo proponemos un enfoque de generación de casos de prueba centrado en la base de datos, donde se prueba el comportamiento de las aplicaciones que la gestionan, comprobando si son capaces de manejar las particularidades de las estructuras definidas en forma adecuada para las distintas capas de la arquitectura (lógica, interfaz de usuario, etc.). Esta propuesta representa el modelo de datos usando *UML Data Modeling Profile*, el cual es extraído automáticamente a partir del esquema relacional de la base de datos mediante técnicas de ingeniería inversa, para luego generar el modelo de pruebas de forma automática usando transformaciones entre modelos. Dicha transformación busca ocurrencias de patrones que, desde el punto de vista del *testing*, representan situaciones de prueba interesantes, y genera casos de prueba siguiendo el estándar *UML Testing Profile*. En este artículo se describe el entorno de trabajo y se presentan, a modo de ejemplo, el diseño de las pruebas para las operaciones de creación y eliminación de instancias.

Keywords: Pruebas de Software, Pruebas basadas en modelos, Pruebas de Sistemas de Información, Datos de Prueba.

1 Introducción

En el desarrollo de software las pruebas juegan un papel muy importante, suponiendo alrededor de la mitad del costo del proyecto. Con el fin de minimizar los costos y aumentar la productividad se apuesta por la automatización de sus tareas, que Meudec [1] clasifica en: (1) tareas administrativas (registro de especificaciones, generación de informes, etc.); (2) tareas mecánicas (ejecución de casos, captura y reejecución, etc.);

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 315-328, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

y (3) tareas de generación de casos de prueba. Mientras que se han conseguido muchos avances en la automatización de las dos primeras, queda aún mucho trabajo por hacer en cuanto a la generación automática de casos, debido en gran parte a la diversidad de entornos y contextos posibles en los que se ejecutarán los casos: en efecto, Ball et al. [2] citaban en 2000 tres líneas de investigación en este sentido, que siguen aún siendo válidas: (1) generación de casos a partir de código fuente para alcanzar algún nivel de cobertura; (2) generación de casos a partir del conocimiento del sistema que tenga el tester y del comportamiento esperado del sistema; y (3) generación a partir de especificaciones formales.

Los sistemas de información (SI) son uno de esos contextos y entornos que hemos mencionado que condicionan la generación automática de casos. En ellos, la base de datos (BD) desempeña un papel fundamental. Por lo general, una misma BD es "atacada" por aplicaciones diversas en función, por ejemplo, del tipo de usuario, el sistema operativo, el entorno de ejecución, el tipo de dispositivo por el que se accede, etc. En muchos casos, existe una correspondencia clara entre el modelo de la BD y la capa de dominio de las aplicaciones que la acceden: de hecho, hay multitud de trabajos cuyo objetivo principal es la propuesta de alguna técnica de ingeniería inversa de la BD para obtener su esquema conceptual y utilizar éste como modelo de la capa de dominio de nuevas aplicaciones de gestión de los datos [3-5]. El esquema conceptual puede obtenerse en forma de un diagrama de clases de *Unified Modeling Languange* (UML): en este caso, se le pueden aplicar técnicas de transformación de modelos para, por ejemplo, generar automáticamente casos de prueba, que es el principal tema de este artículo.

En la última década la utilización de modelos para las diferentes actividades del desarrollo de software se hace cada vez más común. Las pruebas de software no quedan fuera de esta tendencia, destacándose el *Model-Driven Testing* (MDT), que se refiere a pruebas basadas en modelos donde los casos de prueba son generados automáticamente desde artefactos de software mediante transformación de modelos. En este trabajo proponemos usar un enfoque MDT para generar pruebas automáticamente a partir de los metadatos de la BD.

Así, y puesto que la BD es el principal elemento común entre estas aplicaciones, podemos aprovechar su estructura como punto de partida para la construcción de casos de prueba que permitan garantizar la calidad de los programas de acceso y gestión de los SI.

En este trabajo se propone generar pruebas automáticamente para verificar SI que almacenan datos en BD relacionales. Para esto se trabaja sobre un metamodelo basado en UML (extraído automáticamente por mecanismos de ingeniería inversa) que representa el modelo de datos. Luego, para cada subestructura interesante sobre el modelo de datos se generan casos de prueba automáticamente.

Tras describir en la sección 2 los antecedentes y nuestra motivación, se presenta brevemente el enfoque general que se abordará, indicando los distintos estándares que se utilizarán para lograr generalizar la propuesta a otros tipos de SI, y por último se muestran los primeros avances en la generación de pruebas basados en un modelo de datos obtenido a partir de una BD.

2 Contexto de trabajo y motivación

La motivación principal de trabajar en esta temática surgió del trabajo en la industria, en particular probando SI desarrollados con GeneXus©. GeneXus [6] es una herramienta 4GL que permite el desarrollo con un enfoque dirigido por modelos: a partir de la especificación en alto nivel se genera código para distintas plataformas. Esto ha evolucionado en los últimos 20 años de acuerdo al avance de las tecnologías, pasando de sistemas basados en RPG, Windows, Web, y actualmente también para dispositivos móviles. Esta herramienta, de gran implantación en América Latina, Japón, y Estados Unidos, comienza ahora a implantarse en España, en donde ya cuenta con importantes clientes como entidades bancarias. GeneXus tiene su centro en Logroño, donde se realizan anualmente encuentros de usuarios, el último celebrado en marzo de 2011. A finales de 2011 más de 7.000 empresas usan GeneXus, con un total de más de 85.000 usuarios [7]. Esta herramienta permite, manteniendo el modelo de datos, migrar entre las diversas tecnologías. Por ello, para un mismo modelo de datos, puede ser necesario probar la aplicación generada para un ambiente de escritorio, y luego para un ambiente web. Es notorio que lo más importante aquí es poder probar las capas de la aplicación y presentación al usuario, viendo si estas son capaces de manipular correctamente el modelo de datos. Entonces, es interesante generar pruebas centrándose en el modelo de datos que sean válidas para las distintas plataformas que puedan implementarse sobre el mismo.

Desde el 2005 el Centro de Ensayos de Software (CES, www.ces.com.uy) ha trabajado en pruebas de SI en Uruguay y en la región, en muchos casos sobre sistemas desarrollados con esta tecnología. Dado el volumen de trabajo, se vio la conveniencia de crear una herramienta específica para automatizar pruebas de SI construidos con GeneXus, por lo que se creó la empresa Abstracta (www.abstracta.com.uy) que, desde 2007, fabrica el producto GXtest©, que a día de hoy está extendiéndose por la comunidad de usuarios GeneXus, alcanzando ya clientes en 10 países.

Los SI desarrollados con GeneXus son conformes a un metamodelo propietario, no estándar, al cual denominan *Knowledge Base* (KB), o base de conocimiento. En GeneXus se modelan gráficamente las entidades que maneja el SI, sus atributos, relaciones y, a partir de esto, la herramienta genera el modelo de datos y las capas de aplicación y presentación que manejan estos datos, esto es, el soporte a las operaciones de creación, modificación, lectura, eliminación, y listados, para que el usuario final pueda manipular los datos que se pueden almacenar en ese modelo de datos. Luego, GeneXus permite programar, en un lenguaje de alto nivel, procedimientos y eventos que permiten el procesamiento de los datos.

GXtest permite automatizar pruebas al mismo nivel de abstracción que GeneXus: es decir, asociando los artefactos de prueba a los elementos de la KB. Con las herramientas tradicionales se asocian los artefactos de prueba a la aplicación generada, y el problema mayor con esto es que al regenerar la aplicación (ya sea por un cambio en el sistema, o por un cambio de plataforma en la que se desea generar) los artefactos de prueba generalmente deben ser reconstruidos. Con GXtest, sin embargo, se mantienen asociaciones directas entre los artefactos de prueba y los elementos de la KB, absorbiendo el impacto de los cambios.

Una vez que se contaba con la plataforma de automatización de pruebas para GeneXus, se comenzó a investigar acerca de la generación automática de pruebas en

base a la información obtenida en la KB: o sea, a partir del modelo de datos de la aplicación, se generan pruebas que son automáticamente ejecutables sobre el sistema generado.
Vistos los buenos resultados obtenidos en GXtest en la generación de pruebas automática basada en el modelo de datos para SI generados con GeneXus, se ve la posibilidad de generalizar la metodología desarrollada en GXtest hacia otros tipos de SI. Para esto un buen camino a tomar es enlazando con estándares, aprovechando tanto el trabajo existente en la academia y la industria (ya sea sobre modelado, generación de pruebas, etc.), como nuestra propia experiencia (tanto la adquirida en el CES y en Abstracta, como en el Grupo Alarcos de la Universidad de Castilla-La Mancha).

3 Conceptos Previos

Como veremos con detalle en las secciones siguientes, nuestro enfoque de generación automática de casos de prueba consta de tres etapas principales: (1) extracción del modelo de datos; (2) búsqueda de patrones en el modelo y generación de casos de prueba en forma de modelos; (3) generación de casos de prueba ejecutables a partir de los modelos de prueba, que sirvan para probar el SI.

Con objeto de hacerlo lo más generalista posible, trataremos de utilizar, siempre que sea posible, especificaciones estándares para bregar con la representación de los modelos y las transformaciones. Así, nos basaremos en el uso de UML 2.0, que permite su extensión, entre otros mecanismos, mediante perfiles, que se basan en el uso de estereotipos y de valores etiquetados:

- El *UML Data Modeling Profile* (UDMP) [8] es una extensión de UML propuesta por IBM para diseñar BD en UML, pero manteniendo el poder expresivo del modelo de entidad-relación extendido. Define tanto los conceptos a nivel físico y de arquitectura (*Node*, *Tablespace*, *Database*, etc.), hasta los más útiles para el diseño de BD (*Table*, *Column*, etc.). Algunos otros trabajos utilizan también este perfil para el modelado de BD [9-11].
- El *UML Testing Profile* (UTP) [12] es un perfil definido por OMG, que define un lenguaje para diseñar, visualizar, especificar, analizar, construir y documentar artefactos de prueba. Extiende UML con conceptos específicos para las pruebas, agrupándolos en: arquitectura de pruebas, datos de pruebas, comportamiento de pruebas y tiempos de prueba. La arquitectura de las pruebas contiene la definición de todos los conceptos necesarios para realizar las pruebas. El comportamiento de las pruebas especifica las acciones y evaluaciones necesarias para la prueba. El caso de prueba es el concepto principal en el modelo de prueba, y su comportamiento se describe usando el concepto *Behavior* de UML 2.0, diagramas de secuencia, máquinas de estado o diagramas de actividad. En este perfil, un caso de prueba (*test case*) es una operación de un contexto de prueba que especifica cómo un conjunto de componentes cooperan con el sistema bajo prueba para alcanzar el objetivo de prueba, retornando un veredicto. Como es un perfil, se integra sin problemas con UML.
- Respecto de las transformaciones entre modelos, OMG también ha adoptado y publicado el lenguaje de transformación entre modelos QVT (*query/view/transformation*) [13], definido a nivel de metamodelado. Usando

QVT es posible lanzar consultas sobre modelos, y por supuesto, realizar transformaciones entre modelos.

- Además, OMG también ha publicado un lenguaje de transformación modelo-a-texto, llamado MOFM2T [14]. El objetivo es definir un lenguaje que facilite la generación de código fuente o documentación textual a partir de modelos.

En nuestro grupo de investigación tenemos ya cierta experiencia en la utilización de los distintos estándares en diferentes contextos, y también en la búsqueda de patrones en modelos de datos (si bien con un objetivo distinto: la generación automática de servicios web [5]). En este trabajo se aprovechará la experiencia previa en el desarrollo de herramientas y técnicas de reingeniería (por ejemplo, [15]), pruebas ([16]), herramientas comerciales de generación de pruebas (GXtest) y transformaciones de modelos con QVT y UTP [17], con el fin de construir un entorno de pruebas genérico para probar SI.

4 Vista General del Proceso para Generación de Casos de Prueba

Nuestro enfoque consta de las tres etapas que se muestran en la **Fig. 1** y que se describen a continuación:

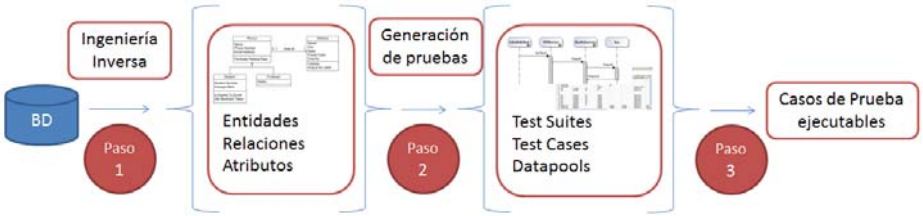


Fig. 1. Esquema general de la metodología

- **Paso 1: Extracción del modelo de datos.** Mediante técnicas y herramientas de ingeniería inversa se obtiene, a partir del esquema físico de la BD, su correspondiente modelo de datos en la forma de un diagrama de clases conforme al UDMP de IBM, en el que quedan representadas las entidades con sus relaciones y atributos. Logramos así representar el modelo de datos en forma independiente de la plataforma sobre la que ejecutará el sistema.
- **Paso 2: Búsqueda de patrones y generación de casos de prueba mediante MDT**. El modelo de datos se procesa mediante técnicas de transformación de modelos. Se buscan, mediante reglas QVT, ciertos patrones interesantes, los cuales no son más que subestructuras del modelo de datos que representan situaciones propicias para ser probadas. Como resultado de las transformaciones, se generan casos de prueba representados con el UTP.
- **Paso 3: Generación de código de prueba.** Por último, tomando como entrada los modelos UTP generados y el modelo de datos, se generan código o scripts de prueba mediante transformaciones MOFM2T.

Respecto del segundo paso, resulta interesante poder dotar al entorno de suficiente capacidad de extensión como para que sea posible definir independientemente nuevos patrones, de manera que se permita incrementar el conocimiento acumulado en generación de pruebas: deseamos que, al ir probando distintos sistemas, se pueda ir enriqueciendo la base de patrones de pruebas, tanto por la propia experiencia y experimentación de este enfoque en sistemas de distinta naturaleza, como de su aplicación a perfiles concretos de usuarios o dominios determinados.

Algo importante a destacar del último paso es que los casos de prueba ejecutables se generan con invocaciones a métodos de una capa de adaptación también generada automáticamente, que luego deberán ser implementados para que accedan a las operaciones correspondientes y sean completamente ejecutables. Esto permite entonces tener pruebas ejecutables siguiendo un enfoque de *Keyword-driven testing* [18]. La mayor ventaja es que se pueden utilizar las mismas pruebas para ejecutarlas contra la aplicación generada para diferentes plataformas.

Hasta el momento tenemos resuelto por completo el primer paso, y estamos trabajando en el segundo y el tercero. Para el primero se trabaja principalmente con las herramientas *Enterprise Architect* (EA, www.sparxsystems.com/products/) y Papyrus en Eclipse (www.eclipse.org/papyrus/), para el modelado de UML. La herramienta IBM Rational Rose Data Modeler, de la compañía que propone el perfil de datos, paradójicamente no lo utiliza, sino que tiene un formato propietario. La ventaja de EA es que da soporte, también gráfico, al UDMP. También hemos extendiendo *Relational Web*, la cual es una herramienta de reingeniería desarrollada previamente en nuestro grupo, adaptándola para obtener el modelo de datos automáticamente a partir del esquema de una BD y guardarlo en formato UDMP siguiendo las técnicas presentadas por Polo et al. [15]. Para el resto del proceso se está trabajando en el diseño del método de generación de pruebas, que deberá garantizar el alcance de ciertos criterios de cobertura sobre el modelo de datos, de lo que nos ocupamos en la siguiente sección.

5 Diseño de Casos de Prueba a partir de Modelos de Datos

Los casos de prueba se generan mediante transformaciones realizadas sobre el modelo de datos. En éste, se identifican situaciones de prueba interesantes (obviamente, desde el punto de vista del testing) mediante la búsqueda de subestructuras del modelo que concuerden con patrones de modelos predefinidos (una situación de prueba interesante, por ejemplo, podría ser la existencia de una asociación *1:** entre dos clases, que podría proceder de la ingeniería inversa de una relación *1:n* entre dos tablas: quizás sea interesante probar el funcionamiento del SI ante la inserción de un registro en la tabla secundaria sin un registro asociado en la tabla principal).

Es importante destacar que las pruebas generadas atacan el sistema completo, basándose en las estructuras de la BD, por lo que se está verificando que el sistema bajo pruebas sea capaz de manejarlas correctamente. Si por ejemplo en el diseño de pruebas un caso de prueba intenta usar un dato que no cumple una regla CHECK que indica que cierto campo entero debe ser mayor que un valor dado (por ejemplo, el campo *Edad* debe ser mayor que 18), claramente esto será bien manejado por el gestor de BD, pero es muy importante ver que el sistema reaccione adecuadamente ante la ex-

cepción que le dará la ejecución de la sentencia SQL, y que el modelo de datos de la BD continúe consistente con el modelo de datos de las capas superiores.

Toda estrategia de generación de pruebas debe ser mensurable. En nuestro caso, también debemos ser capaces de generar casos de prueba que alcancen algún nivel determinado de cobertura, de manera que pueda cuantificarse su completitud. Los criterios son un mecanismo para comprobar, validar y cuantificar que lo que estamos probando realmente "cubre" el modelo, de acuerdo a una teoría de errores dada. A modo de prueba de concepto, nos basaremos en un criterio de cobertura en particular, pero siempre teniendo en cuenta que el entorno no se limita sólo a los criterios aquí planteados, sino que es extensible a nuevos criterios o casos de prueba que se deseen utilizar más adelante.

5.1 Criterios de Cobertura para Diagramas de Clases

En primer lugar, de entre los criterios de cobertura para modelos UML, propuestos por Andrews et al. [19], consideraremos el subconjunto propuesto para diagramas de clases. Puesto que en este trabajo se propone generar los casos de prueba a partir de diagramas de clase, tiene sentido que se mida la completitud de los casos de prueba mediante los siguientes criterios:

- **AEM (*Association-end multiplicity*):** el conjunto de pruebas debe causar que se cree cada par de multiplicidades representativo en las asociaciones del modelo. Así, si existe una asociación cuya multiplicidad es, en un extremo, *p..n*, debería instanciarse la asociación con *p* elementos (valor mínimo), *n* elementos (valor máximo) y con uno o más valores en el intervalo *(p+1, n-1)*.
- **GN (*Generalization*):** el conjunto de pruebas debe conseguir que se cree cada relación de generalización del modelo. Si bien el modelo relacional (del cual procede el diagrama de clases) no posee herencia, seguimos el criterio de Premerlani y Blaha [20], quienes consideran como tales aquellas relaciones de clave externa con cardinalidades *1* a *0:1* entre claves principales.
- **CA (*Class attribute*):** el conjunto de pruebas debe conseguir que se creen conjuntos de valores representativos de los diferentes atributos de cada clase. El conjunto de valores representativos se consigue en tres pasos: (1) crear valores representativos para cada atributo, para lo que puede usarse la técnica de clases de equivalencia; (2) calcular el producto cartesiano de estos valores; (3) eliminar los conjuntos de valores inválidos de acuerdo con el dominio del problema, posibles restricciones que anoten el diagrama, etc.

5.2 Criterios de Cobertura *CRUD*

Además de los criterios anteriores, y dado que todo dato del SI tiene un ciclo de vida que viene determinado por las operaciones que afectan a su persistencia (operaciones *CRUD: Create*, *Read*, *Update*, *Delete)*, también tiene sentido prestar atención a la pruebas de estas operaciones. El ciclo de vida de un dato viene dado por una ocurrencia de una expresión regular del tipo $C{\cdot}R{\cdot}(U_i{\cdot}R)^*{\cdot}D{\cdot}R$ [21], en donde cada U_i representa una de las múltiples operaciones de actualización del objeto. Las operaciones *R (Read)* son simples operaciones de lectura que no alteran el estado del objeto ni el de

sus datos persistentes asociados, por lo que se utiliza para construir el oráculo tras cada operación que sí suponga cambio de estado. Así, por ejemplo, para una *Factura*, un posible ciclo de vida del que excluimos la operación *Delete* sería *crear factura, asignar cliente, escribir fecha, añadir línea 1, añadir línea 2, modificar línea 1, añadir descuento, calcular IVA, calcular total*, en donde todas las operaciones (salvo la primera) son instancias de alguna $Update_i$. Entre cada par de operaciones se añadiría una operación *Read* para comprobar que el objeto se actualiza correctamente.

En trabajos previos [22] construimos una herramienta que genera casos de prueba expandiendo expresiones regulares y combinando con datos de prueba, por lo que sería interesante verificar la efectividad de cruzar ambos enfoques. Sin embargo, la expansión de la expresión regular y su posterior combinación con datos de prueba puede dar lugar a un número casi inmanejable de casos de prueba. Para limitar este número utilizaremos los criterios de cobertura para expresiones regulares de [23], que ya hemos empleado también en otras ocasiones [24].

5.3 Ejemplos de Aplicación

Para ilustrar la aplicación de los criterios de cobertura anteriores, presentaremos el análisis de cómo generar casos de prueba para alcanzar el criterio AEM de los propuestos por Andrews et al., y, luego, de las operaciones interesantes para las entidades de un SI (operaciones CRUD), nos concentramos en los casos particulares de *create* y *delete*.

Para una asociación entre dos tablas, para alcanzar el criterio AEM es necesario considerar las multiplicidades de la relación. Cuando de BD se trata, las relaciones entre dos tablas pueden ser simples (máximo es 1) o múltiples (máximo es N). Las relaciones pueden ser obligatorias u opcionales, en función de si la multiplicidad mínima es 0 o 1. De aquí surgen 16 situaciones que resulta interesante considerar aparte de cada asociación del modelo de datos, las cuales son las combinaciones de las cuatro posibilidades en cada uno de los dos extremos:

- 0..1 (simple y opcional)
- 1 (simple y obligatoria)
- 0..* (múltiple y opcional)
- 1..* (múltiple y obligatoria)

Para cada una de estas situaciones queremos generar casos de prueba y, para el criterio AEM, es necesario probar con los extremos de los rangos de las multiplicidades. Esto nos hace considerar que los valores interesantes son 0, 1 y *, donde * se entiende como una cantidad *N* mayor a 1. Para el caso de relaciones de multiplicidad "muchos", vemos suficiente con considerar un *N* igual a 2, pues es el mínimo valor con el cual se logran establecer múltiples instancias de una tabla asociadas a la de otra (no obstante, se podrían especificar otros valores de *N* en el momento de generar los casos de prueba). Entonces, instanciaremos las relaciones considerando en los extremos multiplicidades de 0, 1 y 2, lo que da un total de 9 situaciones (combinando las 3 posibilidades para cada uno de los 2 extremos). Cada una de estas las queremos aplicar a los 16 casos.

En la **Tabla 1** se muestran los casos diseñados para cubrir el criterio AEM para la operación *Create* para un caso particular, en el que la tabla **A** se asocia con la tabla **B** por medio de una relación [0..* : 1] (**Fig. 2**). Los patrones de prueba generarán conjuntos de prueba de acuerdo a las particularidades de la relación:

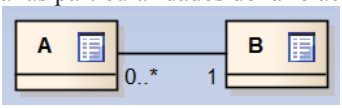
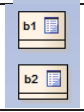
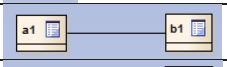
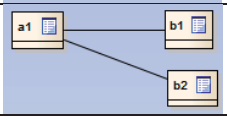


Fig. 2. Ejemplo de relación entre dos tablas

La primera columna de la **Tabla 1** identifica la situación; la segunda y la tercera indican las multiplicidades que se desean crear (o sea, el caso que se está cubriendo), que se muestran gráficamente en la cuarta columna; luego, en las columnas *Resultado Esperado* y *Acción de verificación* se muestra respectivamente el resultado esperado y qué verificar para constatar que la operación fue correcta. Luego, en la **Tabla 2**, se hace el mismo análisis para *Delete*. Para facilitar el análisis se consideraron los 9 casos, y se analizaron los resultados esperados de acuerdo a si cada uno de los extremos es múltiple o no, y si es obligatorio o no. Se parte considerando que no existe ninguna de las instancias a crear (en estos ejemplos no se verifica la unicidad: si se nombran dos instancias diferentes, se supone que no viola ninguna restricción).

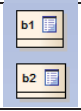
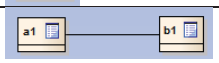
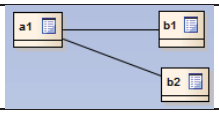
Tabla 1. Diseño de pruebas para la operación *Create*

ID	#A	#B	CREAR	Resultado esperado	Acción de validación
C01	0	0	N/A (no se crea nada)	N/A	N/A
C02	0	1		A no es obligatoria => PASS	Verificar que se creó b1
C03	0	2		A no es obligatoria => PASS	Verificar que se crearon b1 y b2
C04	1	0		B es obligatoria => FAIL	Verificar que no se creó a1
C05	1	1		PASS	Verificar que se creó a1, asociado a b1
C06	1	2		B no es múltiple => FAIL	Verificar que no se pueda asociar a dos elementos
…	…	…	…	…	…

Con objeto de minimizar los tiempos de generación, ejecución y mantenimiento de casos de prueba es interesante mantener, en el *test suite,* el menor número de casos de prueba que, sin embargo, cubran todas estas situaciones interesantes. Además de utilizar, en un futuro, técnicas de reducción de *test suites* como las que hemos aplicado en [25], en nuestro contexto se pueden aprovechar ciertos estados en los que quede la aplicación tras la ejecución de un caso de prueba para probar otro. Por ejemplo, luego de crear un elemento, probar actualizarlo y borrarlo. Entonces, siguiendo este análisis, logramos generar un conjunto de casos de prueba que nos garantizan alcanzar el cu-

brimiento deseado, mientras que minimizamos la ejecución de pruebas gracias al orden de ejecución. Por ejemplo, el resultado de C02 genera un estado en la BD que sirve de entrada para probar D02, así como C05 deja un estado inicial necesario tanto para D05 como D06. Con un breve análisis es fácil ordenar las pruebas de forma tal que los estados iniciales requeridos por algunas pruebas sean generados por otras, determinando así las suites de prueba.

Tabla 2. Diseño de pruebas para la operación *Delete*

ID	#A	#B	ELIMINAR	Resultado esperado	Acción de validación
D01	0	0	N/A (no se borra nada)	N/A	N/A
D02	0	1		A no es obligatoria => PASS	Verificar que se borró b1
D03	0	2		A no es obligatoria => PASS	Verificar que se borraron b1 y b2
D04	1	0		B es obligatoria => N/A	N/A
D05	1	1		Borrar A => PASS	Verificar que se borra a1, y que b1 no tiene nada asociado
D06				Borrar B => FAIL (B es obligatoria)	Verificar que no se borra b1, y sigue asociado a a1
D07	1	2		B no es múltiple => N/A	N/A
…	…	…	…	…	…

Hasta aquí las pruebas están definidas en un alto nivel de abstracción. Luego, a pesar de que no esté explicado en este artículo, al generar el código de pruebas se generarán datos concretos para cada una de las columnas de cada una de las tablas involucradas, respetando los tipos de datos y las restricciones que estén definidas en el modelo de datos (como por ejemplo las reglas *CHECK*).

5.4 Obtención de Casos de Prueba para el UML Testing Profile

Por último, y para comprender la forma en que estos casos de prueba son generados y representados de acuerdo a UTP, se muestra en la **Fig. 3**, como resultado de las transformaciones, los casos de prueba dentro de la arquitectura de pruebas, la que incluye entre otros:

- Un *Test Context*, conteniendo como métodos los casos de prueba (*test cases*) generados;
- Un *Test Component*, responsable de la inicialización de los casos de prueba y de la interacción con el *System Under Test* (*SUT*);

- Un *datapool* por cada entidad probada; cada *datapool* (conjunto de datos de prueba) tiene un *data selector* por cada *test case* para proveer datos específicos a cada uno.

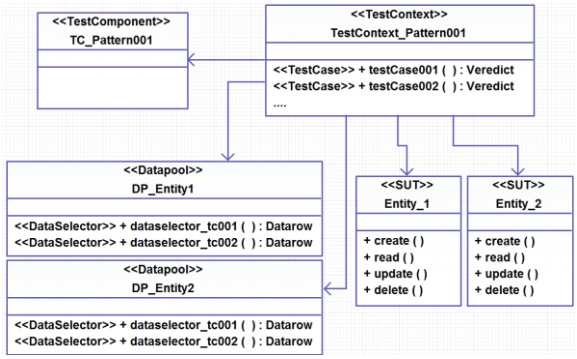


Fig. 3. Arquitectura de pruebas generada en UTP

6 Trabajos Relacionados

Existen varios trabajos que generan datos para sistemas con BD, pero ninguno toma el enfoque dirigido por modelos considerando las operaciones básicas de un SI como proponemos hacer nosotros.

Algunos, por ejemplo, extienden la cobertura basada en líneas de código considerando las particularidades de la interacción del sistema bajo prueba con la BD [26-28], así como las distintas condiciones que se puedan plantear sobre las SQLs que se ejecutan [29, 30]. Así, se generan las instancias de la BD necesarias para cubrir estas situaciones. Tanto en [31] como en [32] se plantea especificar de alguna forma los resultados de las SQL implicadas en la prueba, para de esa forma generar datos para poblar la BD. La propuesta planteada en [33] toma como entradas el esquema de la BD y datos de prueba categorizados dados por el usuario, con lo que genera casos de prueba y estados iniciales para la BD, validando tanto las salidas del sistema como el estado final de la BD. En [34] se generan estados de la BD de acuerdo a las restricciones de integridad del esquema relacional: primero se generan reglas de consistencia de acuerdo a las claves y referencias foráneas, y luego se generan datos que cumplan con ellas.

Existen muchos trabajos que generan pruebas automáticamente a partir de modelos UML [35, 36], pero hasta donde conocemos, solo [37] considera el caso especial de los SI con BD (los cuales tienen particularidades muy importantes a ser tenidas en cuenta). En esa propuesta, Fujiwara et al. proponen generar pruebas considerando un diagrama de clases para representar el modelo de datos, y otro para representar las pantallas. Las relaciones entre tablas son representadas con restricciones OCL, así como también las pre y post condiciones de los métodos a probar. Todo el modelo que especifica el sistema debe ser proporcionado manualmente, y por lo tanto luego debe ser mantenido. Las pruebas que generan están centradas en las restricciones dadas, mientras que en nuestra propuesta veremos que prestamos especial atención al

modelo de datos, y la especificación del sistema es obtenida automáticamente, sin costos de mantenimiento.

7 Conclusiones

En este trabajo se presentó un nuevo enfoque para probar SI que manejan BD, donde se pone especial atención en el cubrimiento de todas las estructuras (simples o compuestas) encontradas en el modelo de datos. Este criterio de adecuación de pruebas tiene en cuenta que en estos sistemas lo importante es que la información esté correctamente almacenada, por lo que se prueban las operaciones que operan sobre estas estructuras conforme a diversos criterios de cobertura.

De esta forma vemos que el enfoque aplicado actualmente a GeneXus, y que es la idea que fue el embrión de este trabajo de investigación (Sección 2), puede ser extendido y aún mejorado para otros tipos de sistemas, generalizando la idea mediante el uso de estándares de la industria, tales como UML, UTP, UDMP, QVT, MOFM2T, etc. Como el entorno de generación de pruebas está casi completamente basado en estándares, puede ser adoptado con casi cualquier herramienta "UML-compliant". Por este motivo, casi no se necesitará implementación de nuevas herramientas para dar soporte a esta metodología.

El artículo ha presentado nuestras primeras ideas y resultados hacia la construcción de un entorno para la generación de pruebas para SI, que irá evolucionando hasta permitir su validación empírica. Por este motivo es importante prestarle atención al trabajo futuro tanto como a las conclusiones presentadas.

8 Trabajo Futuro

A futuro existen muchas líneas de trabajo por donde se plantea continuar.

Por ejemplo, es interesante poder modelar no sólo la estructura del modelo de datos, sino también reglas de negocio que aplican sobre estas estructuras, obtenidas a partir del código fuente, reglas *check*, *deletes* o *updates* en cascada, de los datos del sistema, como para enriquecer las pruebas que de ahí se puedan generar.

Para que las pruebas generadas se puedan ejecutar, como ya se explicó, es necesario indicar (solamente) cómo se accede a cada una de estas operaciones. En el caso de que se esté trabajando en un enfoque de desarrollo dirigido por modelos (MDD) las pruebas podrían ser automáticamente generadas 100%.

Por otro lado, deseamos ir transfiriendo los resultados parciales que se vayan consiguiendo hacia la herramienta GeneXus, lo que nos permitirá, gracias a su amplia implantación, validar los resultados que vayamos obteniendo.

Agradecimientos. Este trabajo ha sido parcialmente financiado por la Agencia Nacional de Investigación e Innovación (ANII, Uruguay), por el proyecto DIMITRI (Desarrollo e Implantación de Metodologías y Tecnologías de Testing, TRA2009_0131) y por el proyecto MAGO/Pegaso (Mejora Avanzada de Procesos Software Globales, TIN2009-13718-C0201).

Referencias

1. Meudec, C., *ATGen: automatic test data generation using constraint logic programming and symbolic execution?*. Software Testing, Verification and Reliability, 2001. **11**(2): p. 81-96.
2. Ball, T., et al., *State generation and automated class testing.* Software Testing Verification and Reliability, 2000. **10**(3): p. 149-170.
3. Alalfi, M.H., J.R. Cordy, and T.R. Dean, *SQL2XMI: Reverse Engineering of UML-ER Diagrams from Relational Database Schemas*, in *Working Conference on Reverse Engineering*. 2008, IEEE Computer Society. p. 187-191.
4. Blaha, M. *A retrospective on industrial database reverse engineering projects. 1*. 2001: IEEE.
5. García Rodríguez de Guzmán, I., *Pressweb: un proceso para la reingeniería de sistemas heredados hacia servicios web*. 2007, UCLM.
6. Artech. *GeneXus*. 1988; Available from: http://www.genexus.com
7. Artech. *GeneXus Facts*. 2012; Available from: www.genexus.com/files/genexus-facts-sheet?es.
8. Gornik, D., *UML Data Modeling Profile*. 2002, IBM, Rational Software.
9. Yin, S. and I. Ray, *Relational database operations modeling with UML*, in *AINA'05: Advanced Information Networking and Applications*. 2005. p. 927-932 vol.1.
10. Zieliriski, K. and T. Szmuc, *Data modeling with UML 2.0.* Software engineering: evolution and emerging technologies, 2006. **130**: p. 63.
11. Sparks, G., *Database modeling in UML*, in *Methods & Tools*. 2001. p. 10-22.
12. OMG. *UML 2.0 Testing Profile Specification*. 2004; Available from: http://utp.omg.org/.
13. OMG, *Meta Object Facility 2.0 Query/View/Transformation Specification*. 2005.
14. OMG, *MOF Model to Text Transformation Language (MOFM2T), 1.0*. 2008.
15. Polo, M., I. García-Rodríguez, and M. Piattini, *An MDA-based approach for database re-engineering.* Journal of Software Maintenance and Evolution: Research and Practice, 2007. **19**(6): p. 383-417.
16. Pérez Lamancha, B. and M. Polo Usaola, *Testing product generation in software product lines using pairwise for features coverage.* Testing Software and Systems, 2010: p. 111-125.
17. Pérez Lamancha, B., et al., *Model-driven Testing - Transformations from Test Models to Test Code*, in *ENASE*. 2011, SciTePress. p. 121-130.
18. Fewster, M. and D. Graham, *Software test automation: effective use of test execution tools*. 1999: ACM Press/Addison-Wesley Publishing Co.
19. Andrews, A., et al., *Test adequacy criteria for UML design models.* Software Testing, Verification and Reliability, 2003. **13**(2): p. 95-127.
20. Premerlani, W.J. and M.R. Blaha. *An approach for reverse engineering of relational databases*. 1993: IEEE.
21. Koomen, T., et al., *TMap Next, for result-driven testing*. 2006: UTN Publishers.
22. Polo, M., S. Tendero, and M. Piattini, *Integrating techniques and tools for testing automation.* Software Testing Verification and Reliability, 2007. **17**(1): p. 3-39.
23. Mariani, L., M. Pezze, and D. Willmor, *Generation of integration tests for self-testing components.* Applying Formal Methods: Testing, Performance, and M/E-Commerce, 2004: p. 337-350.
24. Flores, A. and M. Polo, *Testing-based process for component substitutability.* Software Testing, Verification and Reliability, 2010.
25. Polo Usaola, M., P. Reales Mateo, and B. Pérez Lamancha, *Reduction of Test Suites Using Mutation.* Fundamental Approaches to Software Engineering, 2012: p. 425-438.
26. Haller, K., *White-box testing for database-driven applications: A requirements analysis*. 2009, ACM. p. 13.

27. Shelar, S. and Sdsawarkar, *Database instances generation tool for white-box testing*. 2009. p. 112-116.
28. Emmi, M., R. Majumdar, and K. Sen, *Dynamic test input generation for database applications*, in *ISSTA'07: Software Testing and Analysis*. 2007. p. 151-162.
29. Tuya, J., M.J. Suárez-Cabal, and C. De La Riva, *Full predicate coverage for testing SQL database queries.* Software Testing Verification and Reliability, 2010. **20**(3): p. 237-288.
30. Suárez-Cabal, M.J., C. De La Riva, and J. Tuya, *Populating test databases for testing SQL queries.* IEEE Latin America Transactions, 2010. **8**(2): p. 164-171.
31. Binnig, C., D. Kossmann, and E. Lo, *Multi-RQP - generating test databases for the functional testing of OLTP applications*, in *DBtest'08: International Workshop on Testing Database Systems*. 2008. p. 5.
32. Arasu, A., R. Kaushik, and J. Li, *Data generation using declarative constraints*, in *International conference on Management of data*. 2011, ACM. p. 685-696.
33. Chays, D. and Y. Deng, *Demonstration of AGENDA tool set for testing relational database applications*. 2003, IEEE Computer Society. p. 802-803.
34. Neufeld, A., G. Moerkotte, and P.C. Loekemann, *Generating consistent test data: Restricting the search space by a generator formula.* The VLDB Journal, 1993. **2**(2): p. 173-213.
35. Brucker, A., et al., *A specification-based test case generation method for UML/OCL.* Models in Software Engineering, 2011: p. 334-348.
36. Offutt, J. and A. Abdurazik, *Generating tests from UML specifications.* «UML»'99—The Unified Modeling Language, 1999: p. 76-76.
37. Fujiwara, S., et al., *Test data generation for web application using a UML class diagram with OCL constraints.* Innovations in Systems and Software Engineering, 2011: p. 1-8.

Evaluación de la cobertura en la interacción usuario-base de datos utilizando un enfoque de caja negra

Raquel Blanco, Javier Tuya, Rubén V. Seco

Departamento de Informática, Universidad de Oviedo
Campus de Gijón s/n, 33204 Gijón-Asturias
{rblanco, tuya, valdesruben}@uniovi.es

Resumen. Probar una aplicación de bases de datos es una tarea laboriosa debido a que su comportamiento no sólo depende de los valores suministrados por el usuario a través de un interfaz, sino que también depende de la estructura y la información almacenada en la base de datos. Por ello, durante el diseño de los casos de prueba se debe considerar tanto la interacción con el usuario como la interacción con la base de datos. Además, la estructura de la base de datos puede tener una gran complejidad, lo que dificulta el diseño de datos de prueba de calidad. Este trabajo describe un enfoque basado en la especificación (caja negra) que guía el diseño de los datos de prueba (entradas del usuario y base de datos de prueba) para una aplicación de bases de datos y que evalúa automáticamente la cobertura alcanzada por dichos datos de prueba. Para ello se modela de forma conjunta la estructura de la base de datos y del interfaz del usuario, dando lugar a un modelo llamado Modelo Integrado de Datos (IDM), y se expresa la funcionalidad requerida mediante un conjunto de reglas de negocio, escritas en términos del IDM, que forman el Modelo de Reglas Integrado (IRM). Posteriormente se aplica un criterio de suficiencia basado en MCDC sobre el IRM para derivar automáticamente las situaciones de interés a probar (requisitos de prueba). Finalmente, se evalúa automáticamente la cobertura alcanzada por los datos de prueba diseñados. El enfoque ha sido aplicado a dos aplicaciones de bases de datos y los resultados muestran que permiten diseñar casos de prueba capaces de detectar fallos significativos.

Palabras clave: pruebas sobre bases de datos, pruebas basadas en la especificación, pruebas de caja negra, datos de prueba, evaluación de la cobertura, MCDC, model-based testing

1 Introducción

Las aplicaciones de bases de datos son, en la actualidad, una parte significativa de muchos sistemas comerciales. En este tipo de aplicaciones la lógica de negocio, normalmente implementada mediante una combinación de lenguajes imperativos y del lenguaje SQL [14], interactúa tanto con el usuario a través de un determinado interfaz, como con la base de datos subyacente. Una transacción típica realizada por un usuario comienza mostrando información en el interfaz de usuario que procede de la base de

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 329-342, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

datos. Basándose en dicha información, el usuario introduce nuevos datos en su interfaz. Posteriormente se ejecuta la transacción, teniendo en cuenta no sólo los valores que acaba de suministrar el usuario, sino también los datos almacenados en la base de datos. Tras la ejecución de la transacción, se actualiza la información mostrada en el interfaz de usuario y/o la almacenada en la base de datos.

Probar este tipo de aplicaciones y la interrelación entre el usuario y la base de datos es especialmente importante, dado que un fallo puede producir no sólo que el usuario reciba una salida incorrecta, sino también el daño o la pérdida de información vital almacenada en la base de datos. Además, los datos incorrectos pueden ser la entrada de otros procesos, los cuales pueden funcionar de forma inadecuada y pueden provocar más daño en los datos. Asimismo, los fallos pueden aparecer en el código procedural, en las sentencias SQL que interaccionan con la base de datos, en el esquema de la base de datos o en los datos almacenados en dicha base de datos.

La problemática asociada a las pruebas sobre bases de datos ha centrado la atención de varios investigadores, dando lugar a diversas líneas de investigación. Una de ellas es la definición de criterios de suficiencia, por ejemplo [11, 15, 19, 21, 23, 24, 28, 29]. Estos criterios permiten determinar las situaciones de interés a probar (en adelante, *requisitos de prueba*) y evaluar la calidad de los datos de prueba, de modo que puedan servir de guía para la generación de los datos de prueba. Otra de las líneas se centra en la generación de datos de prueba teniendo en cuenta las consultas SQL que constituyen el código fuente de la aplicación, por ejemplo [2, 4-6, 8-10, 16, 17, 25, 27] o las reglas de negocio que deben estar implementadas en la aplicación, por ejemplo [26].

En la mayoría de los trabajos anteriores, la interacción con el usuario no es tenida en cuenta, por lo que los datos de prueba que se generan no permiten ejercitar en profundidad el comportamiento de la aplicación. Por otro lado los criterios de suficiencia citados están diseñados para tratar con las particularidades del código que accede a la base de datos, por lo que la generación de los datos de prueba está guiada por la implementación de la aplicación (pruebas de caja blanca), en vez de estar dirigida por lo que debería estar implementado de acuerdo a la especificación del sistema (pruebas de caja negra). En consecuencia, los casos de prueba diseñados para cubrir los requisitos de prueba obtenidos al aplicar dichos criterios pueden no detectar fallos cuando la implementación no cumple la especificación del sistema, dado que lo que hace la aplicación lo hace bien, pero no hace lo que debería.

En este trabajo se presenta un enfoque basado en un modelo que permite: (1) derivar automáticamente requisitos de prueba a partir de la especificación de la aplicación de bases de datos, utilizando un criterio basado en MCDC; (2) automatizar la evaluación de la cobertura alcanzada por los datos de prueba generados; (3) proporcionar información sobre los requisitos de prueba no cubiertos, de modo que se puedan generar datos de prueba adicionales para incrementar la cobertura. En nuestro enfoque los datos de prueba están formados por los valores introducidos por el usuario en el interfaz de la aplicación (en adelante, *entrada del usuario*) y por la información almacenada en la base de datos (en adelante, *base de datos de prueba*).

El resto del trabajo se ha organizado en las siguientes secciones: la sección 2 describe nuestro enfoque para automatizar la generación de los requisitos de prueba a

partir de la especificación y la evaluación de la cobertura alcanzada; la sección 3 presenta los resultados de los experimentos utilizando dos aplicaciones de bases de datos. El artículo finaliza con las conclusiones y el trabajo futuro.

2 Planteamiento del problema

Con el propósito de probar una aplicación de bases de datos, consideramos que cada transacción realizada por el usuario constituye una unidad a ser probada (*unidad de prueba*). Estas transacciones implican, como se ha comentado previamente, la selección de información de la base de datos para que sea mostrada en el interfaz de usuario, la introducción de nuevos datos en dicho interfaz en base a la información mostrada, la ejecución de la transacción y la actualización de la información de la base de datos y/o del interfaz de usuario. Para cada unidad de prueba se pueden diseñar varios casos de prueba para cubrir sus requisitos de prueba, y cada uno de ellos está formado por diferentes entradas del usuario y, normalmente, la misma base de datos de prueba (con el fin de reducir el coste de preparación y ejecución de los casos de prueba).

La especificación de las unidades de prueba incluye la descripción de funcionalidad requerida, la estructura de la información manejada y almacenada por la aplicación (la base de datos) y la estructura de la información manejada por el usuario (el interfaz de usuario). En un trabajo previo [3] hemos explicado la fuerte interrelación existente entre estas tres partes y hemos propuesto su modelado de forma conjunta.

Dado que la entrada del usuario y la base de datos de prueba estén estrechamente relacionadas, nuestro enfoque integra el interfaz de usuario y la base de datos en un modelo único denominado Modelo de Datos Integrado (IDM: *Integrated Data Model*), que es descrito en la sección 2.1. El IDM contiene la estructura de los casos de prueba diseñados para las unidades de prueba.

Asimismo, la funcionalidad requerida de cada unidad de prueba se modela mediante un conjunto de reglas de negocio, cada una de las cuales es una sentencia que define o restringe la estructura del negocio o el comportamiento del negocio [12]. Dichas reglas de negocio definen las propiedades que deben cumplir los datos modelados por el IDM, las acciones que se pueden realizar sobre dichos datos y las salidas a dichas acciones. Estas reglas forman el Modelo de Reglas Integrado (IRM: *Integrated Data Model*) que se describe en la sección 2.2

La implementación de nuestro enfoque para derivar los requisitos de prueba y evaluar la cobertura se describe en la sección 2.3 y consta de los siguientes pasos. En primer lugar, se modela la especificación, construyendo el IDM y el IRM. A continuación se derivan los requisitos de prueba a partir de las reglas que forman el IRM, utilizando un criterio de suficiencia basado en Masking-MCDC. Posteriormente los datos de prueba diseñados, teniendo en cuenta la estructura del IDM, son evaluados para determinar los requisitos de prueba cubiertos y no cubiertos. Si algún requisito de prueba todavía no ha sido cubierto, se diseñan nuevos datos de prueba (la automatización de este proceso está fuera del ámbito de este trabajo) y nuevamente se evalúa la cobertura alcanzada.

2.1 Modelo de datos integrado (IDM)

El IDM representa a los datos de prueba de los casos de prueba diseñados para cubrir los requisitos de prueba derivados. Este modelo está compuesto de 3 niveles:

- *Nivel Base de Datos*, que modela la base de datos utilizada por la aplicación y representa a la base de datos de prueba.
- *Nivel UI*, que modela los interfaces de usuario de las unidades de prueba y representa la entrada del usuario de cada caso de prueba. Este nivel se conecta con el nivel *Base de Datos* debido a que los datos introducidos por el usuario están estrechamente relacionados con la información almacenada en la base de datos.
- *Nivel Caso de Prueba*, que representa a los casos de prueba creados para las distintas unidades de prueba. Cada uno de estos casos de prueba está relacionado con una determinada entrada del usuario y comparte la base de datos de prueba con el resto de casos. Este nivel está conectado con el nivel *UI* para identificar la entrada del usuario correspondiente a cada caso de prueba.

Cada nivel es modelado mediante un modelo relacional compuesto de un conjunto de entidades y sus relaciones, llamadas relaciones intra-nivel. Las conexiones entre diferentes niveles son también relaciones entre entidades, llamadas relaciones inter-nivel.

2.2 Modelo de reglas integrado (IRM)

El IRM representa la funcionalidad requerida de cada unidad de prueba mediante un conjunto de reglas de negocio llamadas reglas IRM. Estas reglas pueden imponer condiciones sobre el estado de las entidades y atributos del IDM (reglas de restricciones) o pueden especificar condiciones sobre el estado de dichas entidades y atributos que, en caso de ser ciertas, permiten derivar nuevo conocimiento (reglas de derivación). Para definir las reglas IRM se ha creado un lenguaje denominado IRMDL (Integrated Rules Model Definition Language), el cual está basado en la especificación SBVR (*Semantics of Business Vocabulary and Business Rules*) [18].

Por otro lado, las entidades del IDM referenciadas en las condiciones de las reglas IRM pueden estar relacionadas por diferentes rutas, siendo necesario definir cuál de ellas se va a utilizar con el fin de determinar el contexto en el cual se van a evaluar dichas condiciones. En las reglas IRM se definen dos tipos de contexto: *camino* y *grupo*.

Definición 1: un *camino* P es una secuencia de una o más entidades R_1, R_2, …, R_n del IDM, donde cada par (R_i, R_{i+1}) está directamente conectado mediante una serie de atributos en el predicado $q_{i,i+1}()$:

$$\texttt{Path P is } R_1[q_{1,2}()]R2[q_{2,3}()]\ldots[q_{n-1,n}()]R_n$$

Si el par (R_i, R_{i+1}) está conectado mediante la clave ajena, no es necesario especificar el predicado $q_{i,i+1}()$.

El contexto establecido por P está formado por las tuplas obtenidas del producto Cartesiano de las entidades R_i (i=1..n) que cumplen los predicados $q_{i,i+1}()$.

Definición 2: un *grupo* es un conjunto de tuplas del contexto definido por un camino P que tienen el mismo valor en uno o varios atributos del camino A_1, A_2, …, A_n:

`Frame G is P///`A_1`, `A_2`, …, `A_n

Utilizando entidades y atributos pertenecientes a un determinado contexto, las reglas IRM establecen condiciones que o bien deben ser siempre ciertas, o no deben cumplirse nunca en dicho contexto. Estas condiciones pueden ser de dos tipos: *condiciones de valores* y *condiciones de cuantificación*. A continuación se definen ambos tipos de condiciones, utilizando la notación EBNF [13].

Definición 3: una *condición de valores* define o restringe el valor de las expresiones que incluyen atributos de las entidades pertenecientes al contexto utilizado:

```
condicion_valores ::= expr “is” cuantificador
cuantificador ::= “at least” expr|“at most” expr|“exactly” expr
                |“different to” expr|“like” expr
                |“at least” expr “and at most” expr
```

donde `expr` es una expresión aritmética sobre atributos o una constante.

Definición 4: una *condición de cuantificación* define o restringe el número de tuplas que relacionan dos entidades de un camino, donde una actúa como entidad maestro y otra como entidad detalle. Este tipo de condición expresa el número de tuplas de la entidad detalle que pueden estar relacionadas con una determinad tupla de la entidad maestro:

```
condicion_cuantificacion::= maestro “have” cuantificador detalle
cuantificador ::= “at least” expr|“at most” expr|“exactly” expr
        |“different to” expr|“at least” expr “and at most” expr
```

donde `maestro` y `detalle` son entidades de un camino y `expr` es una expresión aritmética cuya evaluación retorna un número entero mayor o igual a 0.

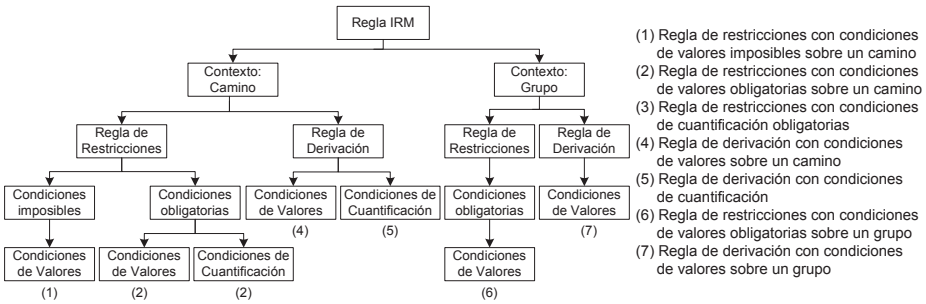


Figura 1. Clasificación de las reglas IRM

La Figura 1 muestra el mapa de las distintas reglas IRM que pueden ser construidas. A continuación se describe cada una de ellas, utilizando la notación EBNF. En las definiciones de estas reglas, `lista_condicion_valores` y `lista_condicion_cuantificacion` son condiciones compuestas mediante los operadores `“and”` y `“or”`, definidas según las reglas de producción `condicion_valores` de la definición 3 y `condicion_cuantificacion` de la definición 4 respectivamente.

- **Reglas de restricciones con condiciones de valores imposibles sobre un camino**, establecen una o varias condiciones que no pueden ser cumplidas simultáneamente en ninguna tupla del contexto definido por un determinado camino:
  `reglaIRM ::= "It is impossible that" lista_condicion_valores`
- **Reglas de restricciones con condiciones de valores obligatorias sobre un camino o sobre un grupo**, establecen una o varias condiciones de valores que deben ser cumplidas por todas las tuplas del contexto definido por un determinado camino o grupo:
  `reglaIRM ::= "It is obligatory that" lista_condicion_valores`
- **Reglas de restricciones con condiciones de cuantificación obligatorias**, establecen una o varias condiciones de cuantificación sobre pares de entidades de un camino que deben ser siempre cumplidas:
  `reglaIRM::="It is obligatory that" lista_condicion_cuantificion`
- **Reglas de derivación con condiciones de valores sobre un camino o sobre un grupo**, permiten definir una serie de acciones para obtener un nuevo conocimiento cuando una o varias condiciones de valores son ciertas en alguna tupla del contexto definido por un determinado camino o un determinado grupo:

```
reglaIRM ::= "if" lista_condicion_valores
             "then" lista_acciones
```

  donde `lista_acciones` es el conjunto de acciones utilizadas para obtener un nuevo conocimiento relativo a los atributos y entidades del IDM.
- **Reglas de derivación con condiciones de cuantificación**, en las cuales las acciones que permiten obtener un nuevo conocimiento son realizadas cuando se cumplen una o varias condiciones de cuantificación:

```
reglaIRM ::= "if" lista_condicion_cuantificacion
             "then" lista_acciones
```

2.3 Implementación

El proceso de derivación de los requisitos de prueba y de evaluación de la cobertura alcanzada ha sido automatizado mediante la herramienta DruidaTest, disponible en http://in2test.lsi.uniovi.es/sqltools/druidatest/.

En primer lugar, DruidaTest construye una base de datos que representa al IDM, en la cual se almacenarán los datos de prueba diseñados. Para ello, la herramienta utiliza el modelo del interfaz de usuario de una unidad de prueba, escrito en un lenguaje similar a SQL denominado IDMDL (*Integrated Data Model Definition Language*), y el esquema de la base de datos utilizada por la aplicación.

Posteriormente DruidaTest deriva los requisitos de prueba aplicando un criterio de suficiencia basado en Masking-MCDC [7] sobre las condiciones establecidas por las reglas IRM y sobre los predicados utilizados para crear el camino que define el contexto en el cual se evalúan dichas condiciones. DruidaTest expresa los requisitos de prueba como consultas SQL que pueden ser ejecutadas contra la base de datos que representa al IDM.

Para automatizar este proceso, DruidaTest transforma las reglas IRM en consultas SQL y aplica sobre ellas un criterio basado en Masking-MCDC denominado SQLFpc [23], que ha sido implementado en el servicio web SQLFpcWS. Este criterio deriva los requisitos de prueba de una consulta SQL también como consultas SQL llamadas *reglas de cobertura*. DruidaTest invoca dicho servicio web con cada consulta SQL obtenida en la transformación de las reglas IRM. Posteriormente modifica y elimina algunas de las reglas de cobertura obtenida y añade otras nuevas para obtener el conjunto final de reglas de cobertura que constituyen la representación ejecutable de los requisitos de prueba de las reglas IRM.

A continuación, DruidaTest muestra al ingeniero de pruebas las reglas de cobertura junto con una descripción en lenguaje natural que indica los datos de prueba que deben ser introducidos en la base de datos que representa al IDM para cubrirlas.

Por último, DruidaTest ejecuta las reglas de cobertura contra dicha base de datos para determinar la cobertura alcanzada (una regla de cobertura se cubre cuando se obtiene al menos una tupla tras su ejecución) y muestra información al ingeniero de pruebas sobre las que no han sido cubiertas, de tal forma que se puedan seguir introduciendo datos de prueba para incrementar la cobertura.

2.4 Ejemplo

Para ilustrar el proceso seguido por el enfoque presentado en este trabajo, consideremos la unidad de prueba "ShoppingCartRecord" de la aplicación web de código abierto Bookstore, que implementa el proceso de actualización/borrado de un pedido existente, formado por un único artículo, por parte de un usuario que ha accedido a dicha aplicación web.

(a) Interfaz de Usuario

```
CREATE TEST ASSIGNMENT
                ShoppingCartRecord{
 CREATE ENTITY UI_order{
  ui_order_id INOUT INT NOT NULL KEY
      REFERENCES Orders.order_id
  ui_login OUTPUT INT NOT NULL
    REFERENCES Members.member_id
  ui_item_id OUTPUT INT NOT NULL
    REFERENCES Items.item_id
  ui_quantity INOUT INT
 }
}
```

(b) Definición del Interfaz de Usuario utilizando el lenguaje IDMDL

Figura 2. Interfaz de usuario de la unidad de prueba "ShoppingCartRecord"

La Figura 2 muestra el interfaz de usuario en la parte (a) (al lado de cada campo aparece un superíndice que indica si es una variable de entrada (I) y/o de salida (O)) y su modelo, utilizando el lenguaje IDMDL, en la parte (b). Este modelo constituye el nivel *UI* del IDM y está formado por una única entidad llamada *UI_order*, que contiene todos los campos del interfaz de usuario. Esta entidad tiene tres relaciones inter-

nivel con entidades del nivel *Base de Datos* del IDM para indicar que el usuario que accede a la unidad de prueba, el pedido a modificar y el artículo de dicho pedido están almacenados en las entidades *Members*, *Orders* e *Items* respectivamente.

El IDM se representa en la Figura 3. El nivel *Base de Datos* está formado por el modelo relacional de la base de datos utilizada por Bookstore. El nivel *UI* está formado por la entidad *UI_order*, la cual es detalle de las entidades del nivel *Base de Datos* dado que cada usuario, pedido y artículo almacenado en la base de datos puede ser utilizado en diversos casos de prueba. Finalmente el nivel *Caso de Prueba* contiene la entidad *TestCase* donde cada caso de prueba es diferenciado. Esta entidad tiene una relación inter-nivel con la unidad de prueba para indicar que se pueden diseñar varios casos de prueba para ella y que cada caso de prueba se corresponde con un determinado pedido a actualizar en el interfaz de usuario.

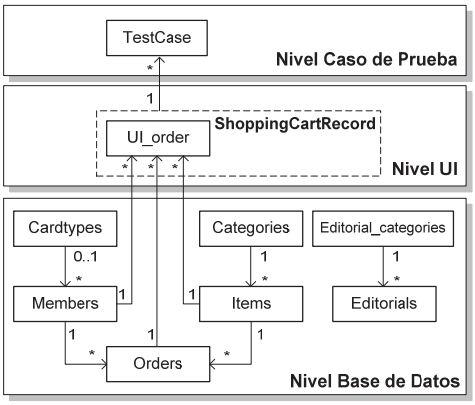


Figura 3. IDM de la unidad de prueba "ShoppingCartRecord"

Para esta unidad de prueba se define una regla IRM que indica que el propietario del pedido a actualizar (atributo *member_id* de la entidad *Orders*) debe ser el usuario que ha accedido a dicha unidad de prueba (atributo *ui_login* de la entidad *UI_order*). Para escribir esta regla, en primer lugar se define el camino seleccionado y posteriormente se especifica la condición de valores que se debe cumplir obligatoriamente en el contexto definido por dicho camino:

```
Path P is UI_order[]Orders
It is obligatory that P.ui_login is exactly P.member_id
```

Los requisitos de prueba que se derivan de esta regla de cobertura se pueden ver en la Tabla 1. Cada fila muestra la regla de cobertura que representa al requisito de prueba y la descripción en lenguaje natural que se muestra al ingeniero de pruebas. Las reglas de cobertura 1 y 2 se derivan de la condición de valores mientras que las reglas 3 y 4 se derivan de la definición del camino.

Tabla 1. Requisitos de prueba para la unidad de prueba "ShoppingCartRecord"

ID	Regla de Cobertura	Descripción en lenguaje natural
1	`SELECT * FROM TestCase INNER JOIN UI_order on (TestCase.tc_id = UI_order.tc_id) INNER JOIN Orders on (ui_order_id = order_id) WHERE NOT (ui_login = member_id)`	A set of joined tables such that: The WHERE condition fulfills: (F) ui_login = member_id is FALSE
2	`SELECT * FROM TestCase INNER JOIN UI_order on (TestCase.tc_id = UI_order.tc_id) INNER JOIN Orders on (ui_order_id = order_id) WHERE ui_login = member_id`	A set of joined tables such that: The WHERE condition fulfills: (T) ui_login = member_id is TRUE
3	`SELECT * FROM TestCase INNER JOIN UI_order on (TestCase.tc_id = UI_order.tc_id) LEFT JOIN Orders on (ui_order_id = order_id) WHERE (order_id is null and ui_order_id is not null)`	summary: [UI_ORDER] left [ORDERS] ON (ui_order_id = order_id) There exist a set of rows joined from tables TESTCASE, UI_ORDER which does not join to any table in ORDERS
4	`SELECT * FROM TestCase RIGHT JOIN UI_order on (TestCase.tc_id = UI_order.tc_id) RIGHT JOIN Orders on (ui_order_id = order_id) WHERE (ui_order_id is null and order_id is not null)`	summary: [UI_ORDER] right [ORDERS] ON (ui_order_id = order_id) There exist some row in table ORDERS which does not join to any table in UI_ORDER

La Figura 4 describe las tuplas que se deben insertar en las entidades de la base de datos que representa al IDM para cubrir las reglas de cobertura (al lado de cada tupla se muestra el número de la regla de cobertura que provoca su inserción). Por ejemplo, la descripción de la regla de cobertura 1 indica que es necesario introducir una tupla en las entidades *TestCase*, *UI_order* y *Orders* para que puedan ser unidas y además se cumpla que el valor de *ui_login* en la tupla insertada en *UI_order* es diferente al valor de *member_id* en la tupla insertada en *Orders*. Asimismo, se tienen que insertar tuplas en las entidades *Items*, *Categories* y *Members* debido a la integridad referencial.

En este ejemplo se han diseñado 3 casos de prueba, cada uno de ellos representado por una tupla en la entidad *TestCase*, la tupla de la entidad *UI_order* relacionada con dicho caso de prueba y las tuplas de las entidades del nivel *Base de Datos*. El caso de prueba con *tc_id = 1* detecta un fallo de seguridad en la implementación de la aplicación, puesto que modificando el número de pedido en la URL es posible acceder y actualizar el pedido de otro usuario.

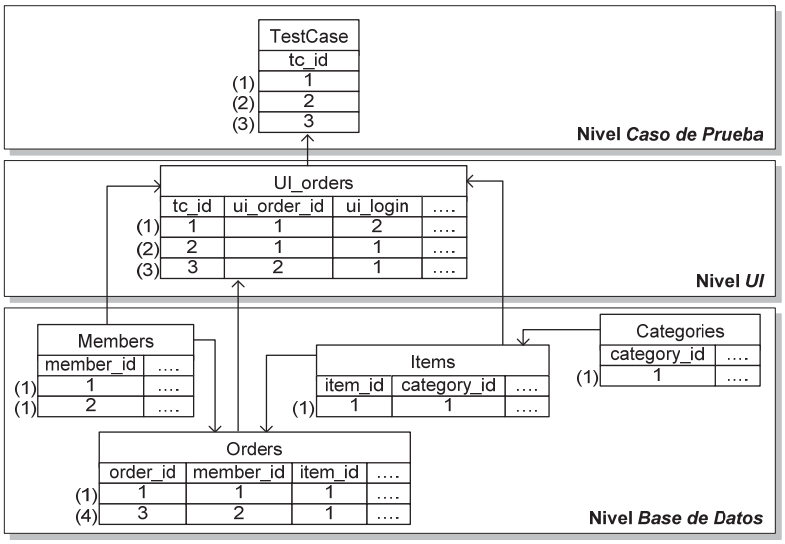


Figura 4. Tuplas insertadas en el IDM para cubrir los requisitos de prueba de la unidad de prueba "ShoppingCartRecord"

3 Caso de estudio

En esta sección se presenta el estudio realizado con dos aplicaciones de bases de datos: el benchmark TPC-C [20], que representa la actividad de una compañía de venta al por mayor (sección 3.1); y FACT, una aplicación para manejo de la facturación de una sociedad (sección 3.2).

En ambos casos se analizó la especificación de la aplicación para modelar los interfaces de usuario de las unidades de prueba y obtener el conjunto de reglas IRM que representaban su funcionalidad. Posteriormente se ejecutó la herramienta DruidaTest para obtener la representación ejecutable de los requisitos de prueba y su descripción en lenguaje natural. La generación de los casos de prueba fue guiada por las descripciones de los requisitos de prueba. A medida que se iban introduciendo datos de prueba en la base de datos que representa al IDM, se evaluaba la cobertura alcanzada hasta que todos los requisitos de prueba fueron cubiertos. En el estudio de cada aplicación se utilizó una misma base de datos de prueba para todos los casos de prueba generados. Por último, se ejecutaron los casos de prueba diseñados y se recogieron los fallos detectados (actualmente la descripción de la salida esperada es realizada por el ingeniero de pruebas)

3.1 Caso de estudio 1: benchmark TCP-C

El benchmark TPC-C consta de 5 unidades de prueba: *New-Order*, *Payment*, *Delivery*, *Order-Status* and *Stock-Level*. De entre las diferentes implementaciones disponibles, se ha seleccionado el benchmark de código abierto BenchmarkSQL [1].

La Tabla 2 muestra, para cada unidad de prueba, el número de reglas IRM y el número de requisitos de prueba generados, así como el número de tuplas insertadas en la base de datos que representa al IDM para cubrirlos (este número acumula las tuplas insertadas en cada nivel del IDM, teniendo en cuenta que para cada caso de prueba se ha insertado una tupla en el nivel *Caso de Prueba*). Asimismo, la Tabla 2 muestra el número de casos de prueba generados, el número total de fallos encontrados y el número de fallos después de eliminar los duplicados. En total se detectaron 13 fallos no duplicados.

Tabla 2. Resultados obtenidos para el benchmark TPC-C

Unidad de Prueba	Reglas IRM	Req. de prueba	Tuplas insertadas			Casos de Prueba	Fallos	Fallos no duplicados
			IDM	Nivel *UI*	Nivel *Base de Datos*			
New-Order	7	38	113	86	17	10	10	5
Payment	6	24	23	9	5	9	7	5
Order-Status	5	19	30	7	16	7	0	0
Delivery	4	20	27	7	13	7	4	2
Stock-Level	4	24	129	8	113	8	3	1
Total:	**26**	**125**	**322**	**117**	**164**	**41**	**24**	**13**

Tras analizar el código fuente de BenchmarkSQL, se detectó que todos los fallos se debían a defectos en el código procedural. De los 13 fallos, 5 estaban causados por defectos relacionados con la validación de las entradas, 3 eran debidos a la incorrecta manipulación de valores nulos, 3 eran provocados por una incorrecta actualización del interfaz de usuario y 2 se debían a un procesamiento de los valores de los datos de prueba que no cumplían la especificación de la aplicación.

3.2 Caso de estudio 2: FACT

FACT es una aplicación para gestionar el pago de los recibos domiciliados emitidos a los socios de una sociedad. Esa aplicación ha sido utilizada para simular el ciclo de vida de la prueba y mantenimiento de una aplicación software en un curso de ingeniería del software [22]. Para este propósito, se inyectaron defectos simples y complejos.

La aplicación consta de 4 unidades de prueba: *generación de recibos*, que gestiona la creación de lotes de recibos correspondientes a un periodo de facturación, incluyendo los recibos de clientes a los que les toca ser facturados y la reclamación de los recibos que han sido devueltos por la entidad bancaria; e*nvío de lotes*, que gestiona el envío de los lotes generados a la entidad bancaria; *recepción de lotes*, que valida de la información remitida por la entidad bancaria relativa a domiciliaciones realizadas; *reclamación de impagados*, que determina los recibos devueltos por la entidad bancaria que pueden ser reclamados en la unidad de prueba *generación de recibos*.

La Tabla 3 muestra, al igual que la Tabla 2, el número de reglas IRM, el número de requisitos de prueba obtenidos, el número de tuplas insertadas en la base de datos que representa al IDM, el número de casos de prueba generados y el número total de fallos encontrados. En el estudio de FACT se han diseñado menos reglas IRM, pero

considerablemente más complejas, que las diseñadas en el estudio del benchmark TPC-C. En total se detectaron 21 fallos con sólo 8 casos de prueba.

Tabla 3. Resultados obtenidos para FACT

Unidad de Prueba	Reglas IRM	Req. de Prueba	Tuplas insertadas			Casos de Prueba	Fallos
			IDM	Nivel *UI*	Nivel *Base de Datos*		
Generación de Recibos	3	68	40	16	21	3	5
Envío de Lotes	1	14	13	5	7	1	1
Recepción de Lotes	3	34	100	35	64	1	8
Reclamación de Impagados	2	18	39	22	14	3	7
Total:	**9**	**134**	**192**	**78**	**106**	**8**	**21**

De entre todos los fallos inyectados, sólo uno no fue detectado, el cual se encuentra en la unidad de prueba generación de recibos. Este fallo se debe a un defecto situado en las acciones que se realizan para generar un nuevo recibo que reclame uno previamente devuelto por la entidad bancaria. Como nuestro enfoque actualmente no maneja las acciones de las reglas IRM, no se generaron requisitos de prueba que permitieran detectar dicho fallo.

4 Conclusiones y trabajo futuro

Este trabajo presenta un enfoque para derivar automáticamente los requisitos de prueba a partir de la especificación de una aplicación de bases de datos y para automatizar la evaluación de la suficiencia de los datos de prueba, teniendo en cuenta el estado de la base de datos y la información introducida en el interfaz de usuario.

La base de datos y el interfaz de usuario se han integrado en un único modelo denominado IDM, el cual contiene la estructura de los casos de prueba de la aplicación. La funcionalidad requerida es expresada mediante un conjunto de reglas de negocio escritas en términos del IDM, denominadas reglas IRM. Sobre dichas reglas IRM se aplica un criterio basado en Masking-MCDC para derivar automáticamente los requisitos de prueba.

Para automatizar la evaluación de la suficiencia de los datos de prueba, el IDM se implementa como una base de datos relacional que almacena dichos datos de prueba (entrada del usuario y base de datos de prueba). Además los requisitos de prueba se expresan como sentencias SQL que pueden ser ejecutadas contra la base de datos que representa al IDM. La ejecución de esas sentencias SQL determina los requisitos de prueba cubiertos y no cubiertos por los datos de prueba. De este modo, nuestro enfoque ayuda a identificar y cubrir situaciones de prueba interesantes tanto para las entradas del usuario como para el estado de la base de datos.

Los resultados muestran que el enfoque presentado permite identificar situaciones de prueba significativas y que los casos de prueba diseñados para cubrirlos son capaces de detectar fallos interesantes.

El trabajo futuro incluye varias líneas. Por un lado, derivar requisitos de prueba a partir de las acciones de las reglas IRM y también a partir de nuevos tipos de reglas

que incrementen la expresividad del IRM. Por otro lado, poblar automáticamente la base de datos que representa al IDM, que es un trabajo en curso. Además, como dicha base de datos puede almacenar la salida de los casos de prueba, se podría automatizar parcialmente la comparación entre la salida actual y la esperada. Asimismo, como las acciones de las reglas IRM expresan el comportamiento de salida de la aplicación, éstas podrían ser también utilizadas como oráculo de prueba.

Además de las líneas anteriores, también se debería realizar una experimentación más extensa para comparar nuestro enfoque con el presente en otros trabajos, con el fin de evaluar su efectividad en la detección de fallos en las aplicaciones de bases de datos.

Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Ciencia e Innovación de España y por los fondos FEDER (TIN2010-20057-C03-01).

Referencias

1. BenchmarkSQL, version 2-3-2, http://sourceforge.net/projects/benchmarksql
2. C. Binnig, D. Kossmann, E. Lo, "MultiRQP - Generating test databases for the functional testing of OLTP applications", Proc. 1st International Workshop on Testing Database Systems (DBTest 08), ACM Press, June 2008.
3. R. Blanco, J. Tuya, R.V. Seco, "Test adequacy evaluation for the user-database interaction: a specification-based approach", Proc. 5th International Conference on Software Testing, Verification and Validation (ICST 2012), IEEE Computer Society, April 2012.
4. D. Chays, S. Dan, P.G. Frankl., F.I. Vokolos, E.J. Weyuker, "A framework for testing database applications", Proc. ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 00), ACM Press, August 2000, pp. 147-157.
5. D. Chays, Y. Deng, P.G. Frankl, S. Dan, F.I. Vokolos, E.J. Weyuker, "An AGENDA for testing relational database applications", Software Testing, Verification and Reliability 14, 1, March 2004, pp. 17-44.
6. D. Chays, J. Shahid, P.G. Frankl, "Query-based test generation for database applications", Proc. 1st International Workshop on Testing Database Systems (DBTest 08), ACM Press, June 2008.
7. J. J. Chilenski, "An investigation of three forms of the modified condition decision coverage (MCDC) criterion", Technical Report DOT/FAA/AR-01/18, U.S. Department of Transportation, Federal Aviation Administration, April 2001.
8. C. de la Riva, M.J. Suárez-Cabal, J. Tuya, "Constraint-based test database generation for SQL queries", Proc. 5th International Workshop on Automation of Software Test (AST 10), ACM Press, May 2010, pp. 67-74.
9. Y. Deng, P. Frankl, D. Chays, "Testing database transactions with AGENDA", Proc. 27th International Conference on Software Engineering (ICSE 05), ACM Press, May 2005, pp. 78-87.
10. M. Emmi, R. Majumdar, K. Sen, "Dynamic Test input generation of database applications", Proc. International Symposium on Software Testing and Analysis (ISSTA 07), ACM Press, July 2007, pp. 151-162.

11. W.G.J. Halfond, A. Orso, "Command-form coverage for testing database applications", Proc. 21st IEEE/ACM International Conference on Automated Software Engineering (ASE 06), IEEE Computer Society, September 2006, pp. 69-80.
12. D. Hay, K. Healy, "Defining Business Rules – what are they really?", Technical Report, The Business Rules Group, Revision 1.3, July 2000.
13. International Standards Organisation. 1996. ISO/IEC 14977 Information technology - Syntactic metalanguage - Extended BNF.
14. International Standards Organisation. 1999. ISO/IEC 9075, Information technology - Database languages – SQL.
15. G.M. Kapfhammer, M.L. Soffa, "A family of test adequacy criteria for database-driven applications", Proc. 9th European Software Engineering Conference held jointly with 11th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (ESEC/FSE 03), ACM Press, September 2003, pp. 98–107.
16. S.A. Khalek, B. Elkarablieh, Y.O. Laleye, A. Khurshid, "Query-aware test Generation using a relational constraint solver", Proc. 23rd IEEE/ACM International Conference on Automated Software Engineering (ASE 08), IEEE Computer Society, September 2008, pp. 238-247.
17. E. Lo, C. Binnig, D. Kossmann, M.T. Özsu, W.K. Hon, "A framework for testing DBMS features", The VLDB Journal 19, 2, April 2010, pp. 203-230.
18. OMG, Semantics of Business Vocabulary and Business Rules Specification, version 1.0, OMG Document Number: formal/2008-01-02, January 2008.
19. M.J. Suárez-Cabal, J. Tuya, "Structural coverage criteria for testing SQL queries", Journal of Universal Computer Science 15, 3, 2009, pp. 584-619.
20. Transaction Processing Performance Council. TPC Benchmark C, Revision 5.10.1, February 2009.
21. J. Tuya, M.J. Suárez-Cabal, C. de la Riva, "Mutating database queries", Information and Software Technology 49, 4, April 2007, pp. 398-417.
22. J. Tuya, C. de la Riva, J. García-Fanjul, "A laboratory exercise in testing database applications", Proc. 7th Workshop on Teaching Software Testing (WTST-7), 2008.
23. J. Tuya, M.J. Suárez-Cabal, C. de la Riva, "Full predicate coverage for testing SQL database queries", Software Testing Verification and Reliability 20, 3, September 2010, pp. 237-288.
24. D. Willmor, S.M. Embury, "Exploring test adequacy for database systems", Proc. 3rd UK Software Testing Research Group, York, UK, 2005, pp. 123-133.
25. D. Willmor, S.M. Embury, "An intensional approach to the specification of test cases for database applications", Proc. 28th International Conference on Software Engineering (ICSE 06), ACM Press, May 2006, pp. 102-111.
26. D. Willmor, S.M. Embury, "Testing the implementation of business rules using intensional database tests", Proc. Testing: Academic & Industrial Conference on Practice and Research Techniques (TAIC-PART 06), IEEE Computer Society, August 2006, pp. 115-126.
27. J. Zhang, C. Xu, S.C. Cheung, "Automatic generation of database instances for white-box testing", Proc. 25th International Computer Software and Applications Conference (COMPSAC 01), IEEE Computer Society, October 2001, pp. 161-165.
28. C. Zhou, P. Frankl, "Inferential checking for mutants modifying database states", Proc. International Conference on Software Testing, Verification and Validation (ICST 2011), IEEE Computer Society, March 2011, pp. 259-268.
29. C. Zhou, P. Frankl, 2011, "JDAMA: Java Database Application Mutation Analyzer", Software Testing, Veriirication and Reliability, 21, 2011, pp. 241-263.

Evolutionary Mutation Testing*

Juan José Domínguez Jiménez, Antonia Estero Botaro, Antonio García Domínguez, and Inmaculada Medina Bulo

Universidad de Cádiz, Escuela Superior de Ingeniería,
C/Chile 1, CP 11002, Cádiz, España,
{juanjose.dominguez, antonia.estero, antonio.garciadominguez, inmaculada.medina}@uca.es

Mutation testing is a fault-based testing technique providing a test criterion: the *mutation score*. This criterion can be used to measure the effectiveness of a test suite in terms of its ability to detect faults. Mutation testing generates *mutants* from the program under test by applying *mutation operators* to it. These mutation operators introduce slight syntactical changes into the program that should be detected by a high-quality test suite.

Most mutant generation systems simply generate all possible mutants. Usually, they include a wide array of mutation operators. Each mutation operator generates a large number of mutants. All mutants need to be run against the test suite to determine whether they can be told apart from the original program in some of its test cases (that is, whether they are *killed* by the test suite or not). The entire process can take a long time for nontrivial programs.

Traditionally, one of the main drawbacks of mutation testing has been the high computational cost involved in the execution of the large number of mutants produced for some programs against their test suites. Several strategies have been described in the literature to address this problem. One of them consists in processing only a subset of all the mutants, using *mutant reduction techniques*. *Mutant sampling* randomly selects a subset of the mutants. *Mutant clustering* classifies mutants into different clusters according to the set of test cases that kills them. *Selective mutation* applies only a subset of the available mutation operators. Recently, *higher-order mutation* has been used to find rare, but valuable, higher-order mutants (HOMs) modeling subtle faults. One of these HOMs can subsume many ordinary (first-order) mutants.

We present a new mutant reduction technique, Evolutionary Mutation Testing (EMT), which tries to generate and execute only some of all the mutants, while preserving testing effectiveness. EMT generates and selects mutants in a single step, reducing the number of mutants to execute by favoring through the fitness function two kinds of mutants: surviving mutants (which have not been killed by the test suite) and difficult to kill mutants (which have been killed by one and only one test case that kills no other mutant). We call them *strong mutants*, and they can be used to improve the quality of the initial test suite. The main steps of EMT are:

* Juan José Domínguez-Jiménez, Antonia Estero-Botaro, Antonio García-Domínguez, Inmaculada Medina-Bulo. Evolutionary mutation testing. *Information & Software Technology*, 53(10): 1108-1123, October 2011

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 343-344, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

1. Generate a set of mutants.
2. Execute these mutants.
3. Evaluate their fitnesses.
4. Apply the evolutionary algorithm to the set of mutants to find stronger mutants.
5. Go to step 2 until the termination condition is met.

Why are we proposing a new technique to generate less mutants? Nowadays, computers are faster than those of a decade ago, so the time needed to execute all the mutants against the test suite is now smaller. At the same time, software development is now performed at a higher level of abstraction. In particular, the WS-BPEL (Web Services Business Process Execution Language) standard has been defined for "programming in the large" composing several Web Services (WS hereafter) into one.

WS-BPEL compositions are usually smaller than traditional programs in the sense that they define the logic of the composition of the external WS or partners, while the bulk of the code is in the WS themselves. Therefore, the number of mutants obtained is comparatively much lower. However, this does not imply a reduction of cost at all. WS-BPEL compositions can require more resources than regular programs, so it is important to reduce the number of mutants. They are typically executed by WS-BPEL engines on top of application servers. Even the deployment and undeployment time devoted to every mutant can be noticeable. Communication between different partners in the composition is achieved by message passing. Timeouts can even be defined for inbound messages.

For these reasons, we implemented the system GAmera to apply EMT to WS-BPEL compositions. GAmera can generate, execute, evaluate and classify the mutants into strong (surviving and difficult to kill mutants) and weak mutants (the rest). GAmera is based on a genetic algorithm (GA). GAs have proved to be an effective heuristic optimization strategy for functions with many local optima that traditional methods find difficult to handle. WS-BPEL mutants are generated by using the 26 mutation operators originally defined for this language by Estero et al.

We have applied GAmera to three WS-BPEL compositions to estimate its effectiveness, comparing it with random selection. The results obtained experimentally show that EMT can select all strong mutants generating 15% less mutants than random selection in over 20% less time for complex compositions. When generating a percentage of all mutants, EMT finds on average more strong mutants than random selection. This has been confirmed to be statistically significant within a 99.9% confidence interval.

Experiments show that EMT has reduced for the three tested compositions the number of mutants required to select those which are useful to derive new test cases that improve the quality of the test suite. The directed search performed by EMT makes it more effective than random selection, especially as compositions become more complex and the search space widens.

Acknowledgements TIN2011-27242 and PR2011-004.

Hacia una propuesta de priorización de casos de pruebas a partir de NDT

Carmen R. Cutilla, Julian A. García-García and Javier J. Gutiérrez

Grupo de investigación IWT2, Universidad de Sevilla, Sevilla, España
{carmen.ruiz, julian.garcia}@iwt2.org, javierj@us.es

Abstract. La importancia de la fase de pruebas ha ido incrementándose de manera exponencial en los últimos tiempos, llegando a considerarse hoy en día como una de las fases más importantes durante el desarrollo de software debido a los riesgos que puede suponer que el hecho de no realizar las pruebas de forma completa o incorrecta. Estos hechos quedan reflejados a la hora de definir la planificación de los proyectos, en los que la planificación se extiende en la fase de pruebas, resultando más costoso el proyecto. Esta situación conlleva la necesidad de estudiar y aplicar nuevas técnicas para ejecutar la fase de pruebas lo más completa posible reduciendo el coste de dicha fase. Entre las técnicas para conseguir este objetivo se encuentra la técnica de priorización de los casos de pruebas. Esta técnica permite generar un conjunto de casos de pruebas idóneo para validar todas las casuísticas del proyecto. En este trabajo de investigación estudiaremos una nueva línea de investigación sobre la priorización de los casos de pruebas.

Keywords: priorización casos pruebas, selección casos pruebas, casos de pruebas, ingeniería guiada por modelos

1 Introducción

La priorización de los casos de pruebas son técnicas utilizadas para disminuir los tiempos de ejecución de las pruebas [11]. Las técnicas de automatización de generación de casos de pruebas actuales son propensas a crear un número elevado de casos de pruebas, entre las que se encuentran duplicidades o casos de pruebas que se encuentran incluidos dentro de otros por estas razones, no es necesario ejecutar todo el conjunto de casos de pruebas que se han generado para validar el sistema.

Las líneas de investigación en las técnicas selección de casos de pruebas se están realizando desde hace décadas (por ejemplo, Agrawal et al. [1993], Chen et al. [1994], Harrold and Soffa [1988], Hartmann and Robson [1990], Leung and White [1990], Ostrand and Weyuker [1988], and Rothermel and Harrold [1997]). Al igual que se ha investigado en técnicas de selección de casos de pruebas, se han abierto líneas de investigación en técnicas de minimización o reducción y priorización de los casos de pruebas, demostrando sus beneficios en sus estudios de casos prácticos.

.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 345-350, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

Sin embargo estas técnicas no han tenido relevancia en la práctica debido a que aún no existe ningún modelo estándar que garantice la calidad de la técnica, inclinándose en los proyectos por la automatización de la ejecución de las pruebas del conjunto completo de casos de pruebas.

La organización de este artículo se indica a continuación. En la sección 2, describiremos el caso práctico en el que hemos comprobado la justificación de una técnica o modelo que nos permita la ordenación o priorización de los casos de pruebas. En la sección 3 indicaremos los objetivos del trabajo de investigación. Las influencias de este trabajo se presentarán en la sección 4. En la sección 5 describiremos el plan de trabajo que seguiremos y por último en la sección 6 indicaremos las conclusiones recogidas.

2 Problema real

Este trabajo de investigación surge tras la participación del grupo de investigación IWT2 (Ingeniería Web y Testing Temprano) [1] en la fase de pruebas del proyecto web AQUA-WS [2]. El objetivo del proyecto AQUA-WS es el desarrollo de una plataforma web para la gestión del ciclo de las aguas de Sevilla, divido en 4 módulos (Gestión de Obras, Gestión de Clientes, Gestión de Redes y Averías y la Oficina Virtual del Cliente), de los cuáles surgieron alrededor de 1800 requisitos funcionales complejos, compuestos por dos o más escenarios. Debido al elevado número de requisitos, el plan de pruebas generado estaba compuesto por alrededor de 7000 casos de pruebas.

Gracias a la automatización en la generación de los casos de pruebas para la elaboración del plan de pruebas [2] [4] se consiguió reducir el número de horas de trabajo en la fase de pruebas. Sin embargo al estar el proyecto AQUA-WS relacionado con funcionalidades económicas y servicios al cliente requería la necesidad de realizar una fase de pruebas completa, validando cada una de las casuísticas del proyecto, requiriendo un número elevado de horas en la fase de pruebas.

Tras el estudio de las diferentes herramientas para la ejecución automática de las pruebas como FitNesse [18], Avignon [19], JMeter [20], Selenium [21], etc, se comprobó que estas herramientas están orientadas a la ejecución de pruebas simples de navegación. En AQUA-WS el objetivo de la fase de pruebas es testear el ciclo de vida de los datos y la interrelación de éstos entre los diferentes módulos. La preparación y control de los datos con las herramientas de testeo automático resultaba más tedioso que la ejecución manual por parte de los técnicos del equipo de pruebas.

Una vez definido el equipo de pruebas para la ejecución de las pruebas, nos encontramos con el problema de tener un conjunto de 7000 casos de pruebas funcionales sin disponer un plan de ejecución de pruebas, es decir, sin tener un orden de ejecución de pruebas o sin conocer si existen casos de pruebas duplicados. ¿Qué caso de pruebas ejecutamos primero?.

3 Objetivos del trabajo

El objetivo principal de este trabajo de investigación es definir una técnica dirigida por modelos que nos permita la priorización de los casos de pruebas de manera automática o semiautomática para aplicar a la metodología NDT (Navigational Development Techniques), de modo que se pueda cubrir las necesidades indicadas en el punto anterior.

Para alcanzar este objetivo final, debemos realizar unos pasos previos de investigación, los cuales enumeramos a continuación:

1. Realizar un estudio comparativo de los métodos y algoritmos de selección y priorización de los casos de pruebas, recogiendo las ventajas e inconvenientes de cada uno de ellos.
2. Analizar de los métodos y algoritmos, con el objetivo de desarrollar un proceso de ingeniería de modelos para la priorización de los casos de pruebas.
3. Definir la automatización de la priorización de los casos de pruebas de manera automática o semiautomática.
4. Validar el proceso definido en un caso práctico con el objetivo de analizar los resultados del proceso.
5. Integrar el modelo definido en la metodología NDT.
6. Ampliar el conjunto de herramientas que ofrece actualmente NDT con la nueva herramienta de priorización de casos de pruebas.

Estos objetivos están basados en trabajos anteriores del grupo de investigación que describiremos en la siguiente sección.

4 Influencias del trabajo

La idea de este trabajo de investigación surgió durante la fase de pruebas del proyecto AQUA-WS, descrito anteriormente en este trabajo, en el que participaba el grupo de investigación IWT2 (Ingeniería Web y Testing Temprano) (Escalona, 2008) [3] Una de las líneas de investigación del grupo es la aplicación de técnicas de testing temprano. Dentro de los recursos más importantes del grupo se encuentra la metodología NDT siendo esta metodología una de las influencias más importantes en este trabajo de investigación. Otras de las influencias a destacar es el trabajo en el que centra su estudio en la fase de pruebas y la generación de pruebas de sistemas a partir de la especificación funcional.

4.1 NDT: Navigational Development Techniques

La metodología NDT, englobada dentro del paradigma MDE (Model-Driven Engineering o Ingeniería Guiada por Modelos) fue propuesta en 2004 para la captura y análisis de requisitos en sistemas web. en su origen, esta metodología proponía una

serie de metamodelos para la fase de análisis y requisitos, y un conjunto de reglas de transformación para obtener los modelos de análisis desde el momento de requisitos. Inicialmente, NDT definía un conjunto de reglas de derivación expresadas bajo el estándar QVT (Query-View-Transformation) (OMG, 2008)[5], que permite generar los modelos de análisis a partir del modelo de requisitos. El estándar QVT define un lenguaje declarativo e imperativo propuesto por la OMG para la transformación de modelos en el contexto de la ingeniería guiada por modelos.

En la actualidad, NDT ha evolucionado hasta ofrecer soporte completo para todo el ciclo de vida del desarrollo software , siendo usado en proyectos de investigación internacionales y en el entorno empresarial. Esta metodología ha ido creciendo en otros aspectos como su enriquecimiento en temas de testing temprano o aspectos de calidad del software.

La metodología NDT está soportada por un conjunto de herramientas, NDT-Suite, apoyadas en la herramienta Enterprise Architect.

4.2 Generación de Pruebas del Sistema a Partir de la Especificación Funcional

El trabajo Generación de Pruebas del Sistema a Partir de la Especificación Funcional [Gutiérrez et al 2011] [4] es otra de las referencias de nuestro. Este trabajo también está basado en la metodología guiada por modelos NDT y los procesos de transformación (Fig. 1) del que se ha generado una herramienta para la generación de pruebas funcionales a partir de los requisitos funcionales. Esta herramienta actualmente está integrada en NDT-Suite y la cual ha sido clave en proyectos como el caso práctico AQUA-WS.

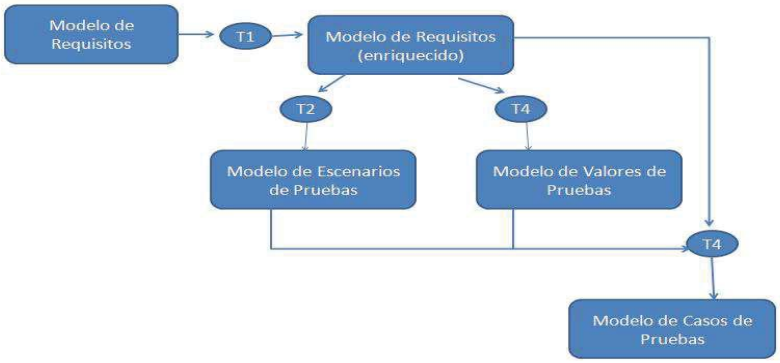


Fig. 1 Proceso de transformación Generación Casos de Pruebas

5 Plan de Trabajo

El plan de trabajo de este trabajo de investigación está constituido por una serie de hitos o tareas para alcanzar los objetivos indicados anteriormente en el apartado.

El primer hito para comenzar el trabajo es realizar un estudio comparativo sistemático de las técnicas de priorización y selección de casos de pruebas. Nuestro estudio seguiremos la guía propuesta por SEG (Software Engineering Group) [6]. Este proceso está compuesto tres actividades principales: planificación, desarrollo de la revisión y publicación de los resultados, compuesta por diferentes tareas.

En la actividad de planificación identificaremos la necesidad de realizar un estudio comparativo, definiendo el protocolo específico que debe seguir el estudio. Hasta el momento en nuestro trabajo de investigación no se han encontrado un estudio comparativo sistemático. El estudio más actualizado (S. Yoo and M. Harman, 2012) [7] realiza la comparación entre las técnicas de minimización, selección y priorización de casos de pruebas, realizando una descripción de cada una de las técnicas. En el estudio comparativo de este trabajo de investigación veremos cómo se comporta cada una de las técnicas sobre el mismo entorno. De este modo comprobamos de una manera clara y concisa las ventajas e inconvenientes de cada una de las técnicas.

Tras este estudio, analizaremos las ventajas e inconvenientes de cada una de las técnicas permitiéndonos tomarlas como referencia a la hora de la definición de la técnica guiada por modelos de nuestro objetivo principal. Una vez definido la técnica continuaremos el trabajo de investigación hasta alcanzar todos los objetivos indicados anteriormente.

6 Conclusiones

A raíz del estudio sistemático que hemos llevado a cabo, se ha comprobado que existen muchas técnicas y metodologías para la priorización de casos de pruebas pero no existe ninguna metodología estándar que garantice la calidad del proceso.

Otro factor clave es que a pesar de existir múltiples tipos de metodologías, no se están aplicando de manera general en proyectos reales o en ocasiones los testers tienen conocimiento de la existencia de éstas técnicas para agilizar su trabajo durante la ejecución de las pruebas.

Mediante la realización de este trabajo de investigación podremos definir una técnica dirigida por modelos, adaptada a la metodología NDT. Además, se pretende desarrollar una herramientas software para automatizar o semi automatizar la técnica que vamos a desarrollar. Esta herramienta se incluirá dentro de la suite de NDT: NDT-Suite.

7 Agradecimientos

Este trabajo de investigación ha sido soportado por el proyecto TEMPROS (TIN 2010-10057-C03-02) y RED CADA (TIN 2010-12313-E) del Ministerio de Ciencia e Innovación, España y el proyecto NTQ-FRAMEWORK de la Junta de Andalucía, España (TIC-5789).

8 REFERENCIAS

1. www.iwt2.org
2. C.R. Cutilla, J.A. García-García, M. Alba, M.J. Escalona, J. Ponce, L. Rodríguez, Aplicación del paradigma MDE para la generación de pruebas funcionales - Experiencia dentro del proyecto AQUA-WS (827-831)., 2011
3. Escalona, M.J., Aragón, G., NDT: A Model-Driven Approach for Web Requirements, IEEE Transactions on Software Engineering, 34(3). pp 370-390, 2008.
4. Gutiérrez, J.J, Mejías M, Escalona, M.J, , Php Generación de Pruebas del Sistema a Partir de la Especificación Funcional. 2011
5. www.omg.org
6. SEG (Software Engineering Group), Guidelines for Performing Systematic Literature Reviews in Software Engineering version 2.3. EBSE Technical Report. EBSE-2007-01. School of Computer Science, 2007.
7. S. Yoo*, M. Harman. Regression testing minimization, selection and prioritization: a survey, Volume 22, Issue 2, pages 67–120, March 2012
8. J.A.N. Lee, Xudong He, A Methodology for Test Selection, 1988
9. Harrold M, Grupta R and Soffa M. "A methodology for controlling the size of a test suite" ACM Transactions on Software Engineering and Methodology, vol 2, nº3, pp.270-285,1993.
10. Harrold M, Grupta R and Soffa M. "A methodology for controlling the size of a test suite" ACM Transactions on Software Engineering and Methodology, vol 2, nº3, pp.270-285,1993.
11. Polo, Macario, Garcia-Rodriguez, I.,Piattini, Mario, Priorización de casos de prueba mediante mutación, 2007
12. Jones, James Harrold, Mary Jean, Test-Suite Reduction and Prioritization for Modified Condition / Decision Coverage Georgia Institute of Technology, 2003
13. Elbaum, Sebastian, Rothermel, Gregg, Kanduri, Satya, Malishevsky, Alexey G., Selecting a Cost-Effective Test Case Prioritization Technique, 2004
14. Mazeiar Salehie, Sen Li, Ladan Tahvildari, Rozita Dara, Shimin Li, Mark Moore: Prioritizing Requirements-Based Regression Test Cases: A Goal-Driven Practice. CSMR 2011: 329-332.
15. A. Srivastava and J. Thiagarajan. Effectively Prioritizing Tests in Development Environment, Proceedings of the 2002 ACM SIGSOFT International Symposium on Software Testing and Analysis, pp. 97-106, Rome, Italy, ACM Press, 2002.
16. Jung-Min Kim and Adam Porter, A History-Based Test Prioritization Technique for Regression Testing in Resource Constrained Environments. Proceedings of the Twenty-fourth International Conference on Software Engineering. Orlando, Fl. May 2002.
17. A Multi-Objective Particle Swarm Optimization for Test Case Selection Based on Functional Requirements Coverage and Execution Efford
18. http://www.fitnesse.org
19. http://sourceforge.net/projects/avignon
20. http://jmeter.apache.org/
21. http://seleniumhq.org

Model-Based Testing in Early Software Development Phases

Silvio Cacace[1] and Tanja E.J. Vos[2]

[1] Dialogues Technology
{silvio.cacace}@dialoguestechnology.nl
[2] Universidad Politecnica de Valencia
{tvos}@pros.upv.es

1 Summary

Despite the clear advantages of using test models, drawing them is not common practice in industry. This is not in the last place because, up to date, no easy to use tool existed that enables the creation of test models and implies less maintenance when requirements change.

In this paper we will discuss a tool that does do these things, yet is very simple, light-weight, easy to learn and does not require experience and or knowledge of difficult formal methods. Basically the tool helps a tester to draw a model, by providing a drawing canvas with a range of shapes and the possibility of connecting them. Secondly, the coverage algorithms underlying the tool, will extract all the combinations of the connections between the elements, and display a minimum set of test cases that can be used for functional or acceptance testing purposes.

2 Introduction

In the eighties, Boris Beizer published his book on blackbox testing techniques [1]. The ideas in this book are simple: draw a model of the expected behaviour of the software system, and cover it. In the book, Beizer advocates that such test suite design activities should start the sooner the better. The presence of the tester in the initial development phases increases the rate of error identification, since ambiguities and unknown issues will be made clear in a very early stage and can help the business to clarify the requirements with the involvement of the stakeholders. Moreover, when testing starts early, the tester can also define the acceptance criteria in that early stage.

Using models or graphs during the early phases of testing has many identified advantages [1, 4, 6]:

- Reviewing a model is more manageable and less error-prone, than reviewing hundreds of test cases.
- Since a model is created from the requirements, it gets easier to trace the model and test cases back to the requirements.
- Models represents the tester's understanding of what the software should do In this way these models can serve as a tool for improving the communication with other stakeholders about the functionalities of the system.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 351-354, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

- Many times, just creating a model from the requirements and showing it to the client will already find important errors in the requirements.
- Using a model to review requirements, takes the 'personal element' out of reviewing. With traditional reviews, people might feel attacked about the quality of their work. A review based upon a model that the tester himself has made, reduces this element since his own model is being questioned.
- If we have a model, test case generation can be automated. If the model changes, the test cases just need to be generated again.
- The scope and coverage of a test suite generated from a model becomes clear.
- A model is easier to maintain than test cases when requirements change.

However, despite the clear advantages of using test models, drawing them is not common practice in industry[2]. This is not in the last place because, up to date, no easy to use tool existed that enables the creation of test models. According to a survey done by Binder last year [2], most Model-Based Testing tools that are around and used by industry are found to be too complex, too formal and have a steep learning curve.

In this paper we will introduce a tool called DTM (Dialogues Testing Method) [5] a very simple, light-weight and easy to learn tool that enables testers to easily create models that can be used for effective communication and automated test case generation. Basically the tool helps the tester to draw a model, by providing a drawing canvas with a range of shapes and the possibility of connecting them. Secondly, the coverage algorithms underlying the tool, will extract all the combinations of the connections between the elements, and display a minimum set of test cases that can be used for functional or acceptance testing purposes. In the following sections we will describe DTM's methodology, the control flow models, test case generation. Finally, Section 6 concludes with future work. The length of this paper only permits illustration of the models and the generated test cases through very simple examples. However, DTM has been used on various real-world software systems and their requirements.

3 Model-Based Testing Methodology

The MBT methodology behind the DTM tool consists of the following steps:

1. Use the requirements documents that are available to create a model. This is an iterative process that consists of:
   (a) Understand the requirements and make a mental model of the system
   (b) Translate the mental model in a flow model and leave open ends for the items that were unclear, inconsistent, ambiguous, and need more explanation from business people, designers or other stakeholders.
   (c) If there are open ends, organize a brainstorm meeting with the necessary stakeholders to close them.
   (d) Update the requirements and update the test model with the obtained information and clarifications.
   (e) If there are no open ends left, review the model with the stakeholders.
   (f) If the model is found complete during the review meeting, continue with step 2. If not, go back to step 1d.

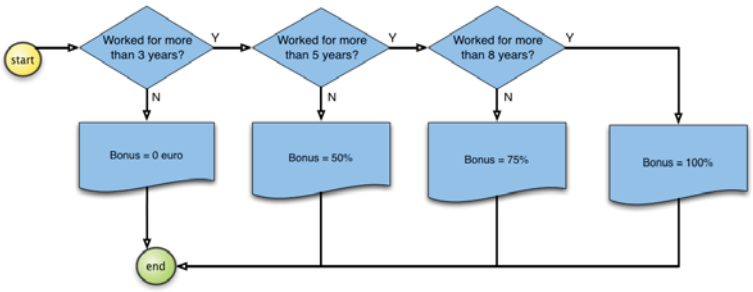


Fig. 1. A simple DTM model for calculating a bonus based on amount of years worked.

2. Generate test cases from the model using a preferred coverage criteria.
3. Execute the tests, debug and fix if defects are found, and update the requirements and the model if necessary.
4. If the requirements do not need to be updated, the test suite can be saved for regression testing.

4 Creating a model with DTM

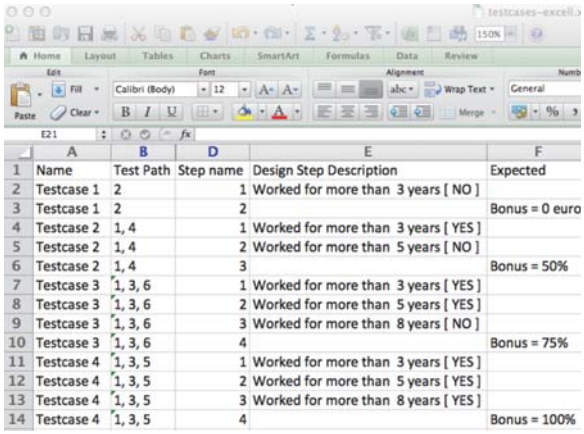
The models used in the DTM tool are control flow graphs. The DTM tool only offers 6 symbols that can be used to draw the test model:

- **start** (a circle with the word start in the centre), is used for indicating the beginning of the control flow.
- **end** (a circle with the word start in the centre), is used for indicating the ending of the control flow.
- **decision** (a diamond) for making binary decisions. The right-arrow is the YES option, the down-arrow is the NO option.
- **action** (an oval), is used to model actions or states.
- **result** (a waved rectangle), is used for storing values in variables or databases.
- **reference** (a pentagon), is used to reference another test model.

An example of a simple model is in Figure 1. The model does not contain actions, nor references and serves merely as an example for the next Section.

5 DTM's Test Case Generation capabilities

When the test model is drawn, abstract test cases can be generated just by one-click. The DTM tool will automatically place numbers beside the arrows coming out from each decision symbol and all the possible test cases will be generated based on a decision coverage algorithm that ensures that each combination of two decisions will be tested at least one time. If test cases on a test model which is not correct, e.g. the model is missing an end symbol, a path is not complete, etc., the tester will be notified by a pop up message. The format of the test cases is in plain text and can be exported to Excel (see Figure 2 for the test cases generated form the model in Figure 1).



	A	B	D	E	F
1	Name	Test Path	Step name	Design Step Description	Expected
2	Testcase 1	2	1	Worked for more than 3 years [NO]	
3	Testcase 1	2	2		Bonus = 0 euro
4	Testcase 2	1, 4	1	Worked for more than 3 years [YES]	
5	Testcase 2	1, 4	2	Worked for more than 5 years [NO]	
6	Testcase 2	1, 4	3		Bonus = 50%
7	Testcase 3	1, 3, 6	1	Worked for more than 3 years [YES]	
8	Testcase 3	1, 3, 6	2	Worked for more than 5 years [YES]	
9	Testcase 3	1, 3, 6	3	Worked for more than 8 years [NO]	
10	Testcase 3	1, 3, 6	4		Bonus = 75%
11	Testcase 4	1, 3, 5	1	Worked for more than 3 years [YES]	
12	Testcase 4	1, 3, 5	2	Worked for more than 5 years [YES]	
13	Testcase 4	1, 3, 5	3	Worked for more than 8 years [YES]	
14	Testcase 4	1, 3, 5	4		Bonus = 100%

Fig. 2. DTM test case descriptions in Excell

6 Conclusions and Future Work

We have briefly introduced the philosophy, the methodology, the control flow models and the test case generation capabilities of the DTM tool. When using models during early phases of a development project, the mere fact of creating a model can reveal inconsistencies, ambiguities and incompleteness in the requirements. This saves time and money, as Boehm [3] indicated in 1976.

Future work that is being contemplated for extending the tool is related to sensitization of the abstract test cases with concrete test data, additional coverage criteria for generating test cases, integration and connection with tools for test management and bugtracking, and empirical evaluations of the usability (i.e. effectiveness, efficiency and subjective satisfaction) of the tool.

References

1. Boris Beizer. *Black-box testing techniques for functional testing*. Wiley, 1995.
2. Robert V. Binder. Model-based testing user survey: Results and analysis, http://www.robertvbinder.com/docs/arts/mbt-user-survey.pdf, 2011.
3. B. W. Boehm. Software engineering. *IEEE Trans. Comp.*, 25(12):1226–1241, 1976.
4. Alan Richardson. Practical experiences with graph based software testing. In *StarEast Software Testing Conference*, 2003.
5. DTM tool. http://www.dtmtool.com/, last visited 27 april 2012.
6. B. Visser and B. Vrenegoor. First model, then test! *Testing Experience*, 17, 2012.

Operadores de Mutación de Cobertura para WS-BPEL 2.0

Antonia Estero Botaro, Juan Boubeta Puig, Valentín Liñeiro Barea, Inmaculada Medina Bulo

Departamento de Ingeniería Informática, Universidad de Cádiz
C/Chile 1, 11002, Cádiz, España
antonia.estero@uca.es,juan.boubeta@uca.es,
valentin.lineirobarea@alum.uca.es,inmaculada.medina@uca.es

Resumen Dada la importancia que en los últimos años están cobrando los servicios web en el ámbito de los procesos de negocio, es imprescindible contar con un soporte de casos de prueba lo suficientemente amplio como para detectar fallos y hacer que se apliquen criterios de cobertura sobre estos servicios. En este contexto está enmarcada la prueba de mutaciones, una técnica de prueba basada en fallos que requiere la definición de un conjunto de operadores de mutación para realizar cambios sintácticos en el programa que se desea probar. En este trabajo se define e implementa, por primera vez, un conjunto de operadores de mutación de cobertura para WS-BPEL 2.0, que aplican los criterios de cobertura definidos dentro del contexto de las pruebas de caja blanca. Además se muestran los resultados experimentales obtenidos al aplicar dichos operadores a varias composiciones WS-BPEL, viendo la aportación de éstos en el proceso de prueba.

Keywords: *prueba de mutaciones, operadores de mutación, criterios de cobertura, servicios web, WS-BPEL*

1. Introducción

En los procesos de negocio, los servicios web o *Web Services* (WS) van a jugar un papel fundamental a la hora de gestionarlos. Casos reales nos podemos encontrar muchos como, por ejemplo, la reserva de un viaje con todo lo que puede conllevar, o la aprobación de un préstamo teniendo en cuenta el riesgo que implica la operación.

Por tanto, es fundamental que tanto los WS como las composiciones que los emplean sean robustos. En este punto entra en juego la *prueba de mutaciones* [14], una técnica de prueba de software basada en fallos que requiere de la definición de un conjunto de operadores de mutación.

Cada uno de estos operadores realiza un cambio sintáctico en el programa a probar, modelando así fallos de programación, o bien, haciendo que se apliquen distintos criterios de cobertura. Así pues, estos operadores pueden ser clasificados en dos categorías: (1) operadores que modelan los errores típicos que suelen

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 355-368, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

cometer los programadores y (2) operadores de cobertura (miden ciertos criterios de cobertura de código). Este trabajo versará sobre esta segunda categoría.

Las nuevas versiones del programa a probar generadas mediante la aplicación de los operadores de mutación se denominan *mutantes*. El objetivo de la prueba de mutaciones es doble: por un lado, permite comprobar la corrección del programa a probar comparando la salida obtenida con la esperada tras ejecutarlo frente a cada caso de prueba[1]; por otro lado, ayuda a generar nuevos casos de prueba que permitan distinguir el programa original de los mutantes generados.

Estero-Botaro et al. definieron en [8] un conjunto de operadores de mutación que modelan los errores típicos que suelen cometer los programadores para el lenguaje de ejecución de procesos de negocio o *Web Services Business Process* 2.0 (WS-BPEL 2.0) [17], y, más tarde, evaluaron cuantitativamente su calidad en [9]. Sin embargo, en ninguno de estos trabajos se estudian ni se definen operadores de cobertura para WS-BPEL.

Este artículo tiene como principal objetivo la definición e implementación de un conjunto de operadores de cobertura para WS-BPEL 2.0, que permitirán generar casos de prueba acordes a criterios de cobertura específicos. Así pues, estos operadores de cobertura completarán el conjunto de operadores para WS-BPEL definido previamente en [8].

Además, se realiza una comparativa con los operadores de cobertura definidos para otros lenguajes estructurados así como un estudio experimental en el que se puede apreciar la información que aportan los nuevos operadores en el proceso de pruebas. Para llevar a cabo este estudio, se ha utilizado la herramienta de mutación para composiciones WS-BPEL 2.0, MuBPEL [10].

WS-BPEL 2.0 es un lenguaje que permite reunir en un solo servicio varios WS existentes, obteniéndose así una *composición de servicios*. Este lenguaje soporta la invocación de WS externos tanto de forma síncrona como asíncrona, así como la concurrencia, lo que hace más complejo el proceso de prueba por las características inherentes que ésta aporta.

El resto del artículo se estructura de la siguiente forma. En la Sección 2 se explican los conceptos más importantes acerca del lenguaje WS-BPEL. Seguidamente, en la Sección 3 se especifica en qué consiste la prueba de mutaciones. En la Sección 4 se presenta una serie de trabajos relacionados que definen operadores de mutación de cobertura para Ada, C y Fortran. A continuación, en la Sección 5 se definen los operadores de cobertura para WS-BPEL 2.0 y en la Sección 6 los operadores de mutación de expresiones que proponemos como operadores que están relacionados con la cobertura para dicho lenguaje. En la Sección 7 se realiza una comparativa de los operadores definidos en las Secciones 5 y 6 con los existentes para otros lenguajes estructurados presentados en la Sección 4. En la Sección 8 se muestran los resultados del estudio experimental realizado para ver la aportación de los operadores definidos en este trabajo. Por último, en la Sección 9 se presentan las conclusiones y el trabajo futuro.

[1] Un caso de prueba es una posible entrada para un programa junto con la salida esperada y las condiciones de ejecución [IEEE Standard 610 (1990)].

2. El Lenguaje WS-BPEL 2.0

WS-BPEL es un lenguaje basado en XML [5] que permite especificar el comportamiento de un proceso de negocio basado en interacciones con WS. La estructura de un proceso WS-BPEL se divide en cuatro secciones:

1. Definición de relaciones con los socios externos, que son el cliente que utiliza el proceso de negocio y los WS a los que llama el proceso.
2. Definición de las variables que emplea el proceso, basada en *Web Services Description Language* (WSDL) [6] y *XML Schema* (XSD) [3].
3. Definición de los distintos tipos de manejadores que puede utilizar el proceso.
4. Descripción del comportamiento del proceso de negocio; esto se logra a través de las actividades que proporciona el lenguaje.

Todos los elementos comentados anteriormente son globales por defecto. Sin embargo, también existe la posibilidad de declararlos de forma local mediante el contenedor *scope*, que permite dividir el proceso de negocio en diferentes ámbitos.

Los principales elementos constructivos de un proceso WS-BPEL son las actividades, que pueden ser de dos tipos: básicas y estructuradas. Las actividades básicas son las que realizan una determinada labor (recepción de un mensaje de un socio externo, manipulación de datos, etc.). Las actividades estructuradas pueden contener otras actividades y definen la lógica de negocio.

Cada actividad se representa mediante un elemento XML. A las actividades pueden asociarse un conjunto de atributos y un conjunto de contenedores. Estos últimos pueden incluir diferentes elementos, que a su vez pueden tener atributos asociados.

Además, WS-BPEL permite realizar acciones en paralelo y de forma sincronizada. Por ejemplo, la actividad `flow` permite ejecutar un conjunto de actividades concurrentemente especificando las condiciones de sincronización entre ellas. A continuación, mostramos un ejemplo de esta actividad:

```
<flow> ← Actividad estructurada
  <links> ← Contenedor
    <link name="comprobarVuelo-A-reservarVuelo" ← Atributo /> ← Elemento
  </links>
  <invoke name="comprobarVuelo" ... > ← Actividad básica
    <sources> ← Contenedor
      <source linkName="comprobarVuelo-A-reservarVuelo" ← Atributo /> ← Elemento
    </sources>
  </invoke>
  <invoke name="comprobarHotel" ... />
  <invoke name="comprobarAlquilerCoche" ... />
  <invoke name="reservarVuelo" ... >
    <targets> ← Contenedor
      <target linkName="comprobarVuelo-A-reservarVuelo" /> ← Elemento
    </targets>
  </invoke>
</flow>
```

Como puede observarse, la actividad `flow` invoca a tres WS en paralelo, `comprobarVuelo`, `comprobarHotel` y `comprobarAlquilerCoche`. Además, existe

otro WS, `reservarVuelo`, que solo se invocará si se completa `comprobarVuelo`. Esta sincronización entre actividades se consigue estableciendo un enlace o *link*; por lo que la actividad objetivo del enlace se ejecutará solo si se ha completado la actividad fuente de ese enlace.

En WS-BPEL pueden utilizarse distintos lenguajes de expresiones. Todos los motores WS-BPEL estándar soportan *XML Path Language* 1.1 (XPath) [2]. XPath es un lenguaje declarativo que permite realizar consultas sobre documentos XML y que dispone de los operadores aritméticos, relacionales y lógicos, con una sintaxis similar a la de los lenguajes tradicionales.

3. Prueba de Mutaciones

La prueba de mutaciones genera a partir de un programa original un gran número de programas denominados *mutantes*, que contienen una única diferencia con respecto al programa original. Los mutantes se generan aplicando al código fuente un conjunto de reglas definidas previamente, los operadores de mutación, que introducen pequeños cambios sintácticos basados en los errores que suelen cometer habitualmente los programadores, o bien pretenden forzar ciertos criterios de cobertura del código. Estos operadores introducen cambios en el programa a probar manteniendo su validez sintáctica.

Una vez generados, los mutantes se ejecutan sobre los casos de prueba; si la salida que produce el mutante es diferente de la que produce el programa original sobre un determinado caso de prueba, se dice que el mutante está *muerto*. En ocasiones, aparecen mutantes que siempre producen la misma salida que el programa original, por lo que no va a existir ningún caso de prueba que permita matarlos; éstos se denominan *mutantes equivalentes*. La calidad de un conjunto de casos de prueba se calcula mediante la *puntuación de mutación*, el cociente entre el número de mutantes muertos y el número de mutantes no equivalentes.

4. Trabajos Relacionados

En la literatura existe una gran cantidad de artículos sobre prueba de mutaciones, revisados por Jia y Harman en [12]. Nosotros nos centraremos en los que analizan y definen operadores de mutación de cobertura para los lenguajes estructurados más conocidos. En la Tabla 1 se presentan estos operadores de cobertura junto con otros que, aunque modelan los errores típicos que suelen cometer los programadores, también permiten medir ciertos criterios de cobertura.

En cuanto a los operadores de mutación definidos para los lenguajes estructurados, Agrawal et al. definen un conjunto de 77 operadores para C [1], que han sido implementados en la herramienta Proteum [7], y los comparan con los operadores de mutación definidos para Fortran. Por otro lado, King y Offutt definen 22 operadores de mutación para Fortran y los implementan en la herramienta Mothra [13]. Además, Offutt et al. definen 65 operadores de mutación para Ada [15] y realizan una comparativa entre los operadores definidos para Ada, C y Fortran.

Tabla 1. Operadores de mutación de cobertura de algunos lenguajes estructurados

Operador	Lenguaje	Referencia	Operador	Lenguaje	Referencia
ABS	Fortran	[13]	Oior	C	[1]
CCO	Ada	[15]	ROR	Fortran	[13]
CDC	Ada	[15]	SAN	Fortran	[13]
CDE	Ada	[15]	SEE	Ada	[15]
EAI	Ada	[15]	UOI	Fortran	[13]
ENI	Ada	[15]	Uuor	C	[1]
EUI	Ada	[15]	VDTR	C	[1]
LCR	Fortran	[13]			

5. Definición de Operadores de Cobertura para WS-BPEL 2.0

En esta sección se proponen y definen unos operadores de mutación de cobertura para WS-BPEL, cuyas descripciones se resumen en la Tabla 2. Estos operadores completarán el conjunto de operadores de mutación definidos por Estero-Botaro et al. [8], que modelan los fallos cometidos por los programadores al escribir código WS-BPEL.

Tabla 2. Operadores de mutación de cobertura para WS-BPEL 2.0

Operador	Descripción
CFA	Sustituye una actividad por la actividad `exit`.
CDE	Sustituye una decisión por `true()` o `false()`.
CCO	Sustituye una condición por `true()` o `false()`.
CDC	Sustituye una decisión o condición por `true()` o `false()`.

5.1. Cobertura de Sentencias

El criterio de cobertura de sentencias consiste en obtener casos de prueba que permitan la ejecución de todas las sentencias que componen el programa.

CFA es el operador de cobertura de sentencias definido para WS-BPEL. Este operador sustituye una actividad por la actividad `exit`, la cual finaliza la ejecución de la composición de manera abrupta. Veamos un ejemplo:

Programa original:

```
<sequence>
  <if>...</if>
</sequence>
```

CFA-01:

```
<sequence>
  <exit>
</sequence>
```

Como podemos ver en el mutante que se ha generado (denominado CFA-01) al aplicar el operador CFA al programa original, se ha sustituido la actividad `if` por la actividad `exit`, para asegurar la ejecución hasta ese punto.

Dado que el mutante acaba su ejecución de manera anormal, cualquier caso de prueba que pase por ese punto matará al mutante obtenido con CFA.

5.2. Cobertura de Decisión

La cobertura de decisión o de ramas tiene como objetivo principal la generación de un conjunto de casos de prueba en los que el flujo del programa pase por cada rama del mismo, es decir, en los que cada decisión sea tenida en cuenta.

CDE es el operador de cobertura de decisión que definimos para WS-BPEL. Este sustituye una decisión de la composición por el valor *true* o *false*. Como utilizamos el lenguaje de expresiones XPath, se utilizarán las funciones que proporcionan estos valores: *true()* o *false()*, respectivamente.

Un ejemplo de aplicación del operador CDE es el siguiente:

Programa original:

```
<sequence>
  <if>
    <condition>
    ($a > 1 and $b = 0)
    </condition>
    ...
  </if>
</sequence>
```

CDE-01:

```
<sequence>
  <if>
    <condition>
    (true())
    </condition>
    ...
  </if>
</sequence>
```

CDE-02:

```
<sequence>
  <if>
    <condition>
    (false())
    </condition>
    ...
  </if>
</sequence>
```

5.3. Cobertura de Condición

El criterio de cobertura de condición se encarga de asegurar que dentro de cada decisión o rama, cada condición toma todos los valores posibles una vez como mínimo.

Así pues, el operador CCO sustituye cada una de las condiciones existentes en la composición por *true()* y *false()*, es decir, por los valores que pueden tomar. A continuación podemos ver un ejemplo, donde CCO-01, CCO-02, CCO-03 y CCO-04 son los mutantes obtenidos tras aplicar CCO:

Programa original:

```
<sequence>
  <if>
    <condition>
    ($a > 1 and $b = 0)
    </condition>
    ...
  </if>
</sequence>
```

CCO-01:

```
<sequence>
  <if>
    <condition>
    (true() and $b = 0)
    </condition>
    ...
  </if>
</sequence>
```

CCO-02:

```
<sequence>
  <if>
    <condition>
    (false() and $b = 0)
    </condition>
    ...
  </if>
</sequence>
```

CCO-03:

```
<sequence>
  <if>
    <condition>
    ($a > 1 and true())
    </condition>
    ...
  </if>
</sequence>
```

CCO-04:

```
<sequence>
  <if>
    <condition>
    ($a > 1 and false())
    </condition>
    ...
  </if>
</sequence>
```

5.4. Cobertura de Decisión/Condición

En algunas situaciones necesitamos que se cumplan a la vez tanto los criterios de cobertura de ramas como de condiciones, para paliar las lagunas que cada uno puede tener en algún contexto determinado. Para ello se define el criterio de cobertura de decisión/condición.

De esta manera, el operador CDC cambia todas las decisiones y condiciones del programa por todos los valores que pueden tomar, como establecen los criterios que engloba.

Tomando el mismo programa original presentado anteriormente, este operador producirá los mismos mutantes generados al aplicar el operador de cobertura de decisión (mutantes CDE-01 y CDE-02) y el operador de cobertura de condición (mutantes CCO-01, CCO-02, CCO-03 y CCO-04). Por tanto, la aplicación del operador CDC subsume la de los operadores CCO y CDE.

Por ello, no es recomendable aplicar todos los operadores de cobertura a la vez, ya que se generarían muchos mutantes repetidos.

6. Operadores de WS-BPEL Relacionados con la Cobertura

En ciertos contextos, los operadores de mutación de expresiones pueden servir para que se aplique algún criterio de cobertura, a pesar de que no cumpla ninguno en concreto. Por ejemplo, la adición del menos unario en las expresiones aritméticas puede hacer que el flujo del programa pase o no pase por un conjunto determinado de instrucciones, obteniéndose así cierta cobertura de ramas.

Algunos de estos operadores de mutación de expresiones, que modelan fallos que podrían cometerse al generar una composición WS-BPEL, no se definieron en el trabajo [8] argumentándose que dichas composiciones no se suelen escribir de forma directa, sino mediante herramientas gráficas. Sin embargo, los autores no tuvieron en cuenta que normalmente las expresiones XPath introducidas en composiciones WS-BPEL sí que suelen ser escritas a mano, lo que puede ocasionar fallos accidentales en su escritura.

Así pues, en este trabajo se añaden dichos operadores para comprobar la aportación que conllevan a la técnica de prueba de mutaciones. En la Tabla 3 se presenta una breve descripción de estos operadores de mutación de expresiones que están relacionados con la cobertura para WS-BPEL 2.0.

Tabla 3. Operadores de mutación relacionados con la cobertura para WS-BPEL 2.0

Operador	Descripción
EIU	Inserta el menos unario en una expresión aritmética.
EIN	Inserta la negación en una expresión lógica.
EAP	Inserta el valor absoluto positivo en una expresión aritmética.
EAN	Inserta el valor absoluto negativo en una expresión aritmética.

6.1. Inserción del Menos Unario

El operador EIU inserta el operador menos unario a una de las expresiones aritméticas de primer nivel, es decir, en el caso de tener una expresión formada por subexpresiones, solo se aplicaría a la expresión completa, no a las anidadas.

Veamos a continuación un ejemplo:

Programa original:

```
<assign>
  <copy>
    <from>$i * $i</from>
    <to variable="Resultado" />
  </copy>
</assign>
```

EIU-01:

```
<assign>
  <copy>
    <from>-($i * $i)</from>
    <to variable="Resultado" />
  </copy>
</assign>
```

Como puede observarse, se ha añadido el menos unario a la expresión XPath presente en la actividad `from`.

6.2. Inserción de la Negación Lógica

El operador EIN añade el operador de negación a una expresión lógica, sin tener en cuenta el posible anidamiento de la misma. A continuación podemos ver un ejemplo de aplicación:

Programa original:

```
<if>
  <condition>$i < 35</condition>
  ...
</if>
```

EIN-01:

```
<if>
  <condition>not($i < 35)</condition>
  ...
</if>
```

6.3. Inserción del Valor Absoluto Positivo

El operador EAP inserta el valor absoluto positivo[2] a una de las expresiones aritméticas de primer nivel de la composición que se desee mutar, de manera análoga a los operadores anteriores.

A continuación, se presenta un ejemplo de aplicación del operador EAP:

[2] Dado que la versión de ActiveBPEL empleada no soporta XPath 2.0, la función *abs()* se implementa de la siguiente forma: $abs(x) = x \cdot ((x \geq 0) - (x < 0))$

Programa original:

```
<assign>
  <copy>
    <from>$i - $j</from>
    <to variable="Resultado" />
  </copy>
</assign>
```

EAP-01:

```
<assign>
  <copy>
    <from>
    ((($i - $j) * (((($i - $j) >= 0) -
    ((($i - $j) < 0)))
    </from>
    <to variable="Resultado" />
  </copy>
</assign>
```

En este ejemplo se ha tomado el valor absoluto de la expresión perteneciente al `from` hijo de la actividad `copy`.

6.4. Inserción del Valor Absoluto Negativo

El operador EAN inserta el valor absoluto negativo a una de las expresiones aritméticas presentes en la composición, excepto en las anidadas.

Veamos la aplicación del operador EAN:

Programa original:

```
<assign>
  <copy>
    <from>$i - $j</from>
    <to variable="Resultado" />
  </copy>
</assign>
```

EAN-01:

```
<assign>
  <copy>
    <from>
    -((($i - $j) * (((($i - $j) >= 0) -
    ((($i - $j) < 0)))
    </from>
    <to variable="Resultado" />
  </copy>
</assign>
```

A partir de este programa original, que es el mismo que se tomó para aplicar el valor absoluto positivo, se genera el mutante en el que se ha insertado, esta vez, el valor absoluto negativo.

7. Comparativa de los Operadores de Cobertura Definidos para WS-BPEL y otros Lenguajes

En esta sección se realiza una comparativa de los operadores de cobertura definidos para los lenguajes estructurados más populares, como es el caso de C, Ada y Fortran, con los operadores de cobertura definidos para WS-BPEL en la Sección 5. Esta comparativa amplía la que se describe en [4], donde se compara únicamente los operadores que modelan los fallos que pueden cometer los programadores al escribir programas WS-BPEL. Por tanto, completamos el estudio que compara los operadores de mutación definidos para WS-BPEL con los definidos para otros lenguajes, teniéndose en cuenta tanto los operadores

que modelan fallos cometidos por los programadores como los que miden ciertos criterios de cobertura.

En la Tabla 4 se resume la comparativa de los operadores que hemos definido para WS-BPEL 2.0 en este trabajo con los existentes para otros lenguajes de programación. Los operadores definidos para otros lenguajes que son equivalentes a los definidos para WS-BPEL se han clasificado en dos categorías: operadores de cobertura y los que están relacionados con la cobertura (aunque son operadores que modelan fallos). Todos estos operadores se han separado, a su vez, en dos grupos: los *operadores similares*, aquellos que necesitan ser redefinidos con pequeños cambios sintácticos para que sean equivalentes a los de WS-BPEL, y los *operadores idénticos* (aparecen marcados en negrita en la tabla), que realizan exactamente la misma mutación que los definidos para WS-BPEL.

Como puede observarse en la Tabla 4, todos los operadores de mutación de cobertura y los relacionados con ésta para Ada son idénticos a los de WS-BPEL, a excepción del operador EIN que no es equivalente a ninguno de Ada.

Sin embargo, ocurre el caso contrario para los operadores definidos para Fortran y C, es decir, todos son similares a los de WS-BPEL, excepto uno de ellos que es idéntico: SAN para Fortran y STRP para C son idénticos a CFA para WS-BPEL.

Los operadores para Fortran similares a los definidos para WS-BPEL son los siguientes. LCR y ROR son similares a ELL y ERR para WS-BPEL, respectivamente, tal y como se comenta en [4]; además, para aplicar los criterios de cobertura de decisión, condición y decisión/condición en Fortran, es necesario que sean aplicados conjuntamente, lo que difiere de nuestra definición. El operador UOI es similar a EIU para WS-BPEL porque además de insertar el menos unario también lo elimina. Finalmente, el operador ABS aúna las mutaciones que realizan los operadores de inserción del valor absoluto positivo y negativo en WS-BPEL: EAP y EAN, respectivamente.

Cabe destacar que existen menos operadores de mutación para C que sean similares a los de WS-BPEL. Por un lado, la definición del operador Oior es más completa que la de CDE para WS-BPEL. Por otro lado, Uuor no solo inserta el menos unario como hace EIU para WS-BPEL, sino que además inserta otros operadores unarios como el más o el valor absoluto. Finalmente, VDTR es análogo al operador ABS para Fortran y, por tanto, es similar a los operadores EAP y EAN para WS-BPEL.

8. Estudio Experimental

Esta sección describe el estudio experimental realizado para comprobar si los nuevos operadores de mutación definidos aportan información relevante al proceso de prueba, con respecto a la que aportaba el conjunto inicial de operadores definidos para WS-BPEL en [8]. Para ello hemos utilizado cuatro composiciones WS-BPEL: *CompraVenta* (CV), *Préstamo* (P), *ServicioEnvío* (SE) y *SumaCuadrados* (SC). La composición *CompraVenta* (174 LOC) es una versión ampliada de la composición *MarketPlace* que proporciona como ejemplo ActiveVOS, la

Tabla 4. Equivalencias entre los operadores de cobertura definidos para WS-BPEL 2.0 y otros lenguajes estructurados

Categoría	WS-BPEL 2.0	Ada	Fortran	C
Operadores de cobertura	CFA	**SEE**	**SAN**	**STRP**
	CDE	**CDE**	LCR	Oior
	CCO	**CCO**	ROR	–
	CDC	**CDC**	LCR, ROR LCR	–
Operadores relacionados con la cobertura	EIN	–	–	–
	EIU	**EUI**	UOI	Uuor
	EAP	**EAI**	ABS	VDTR
	EAN	**ENI**	ABS	VDTR

empresa que desarrolla ActiveBPEL. La composición *Préstamo* (117 LOC) es una mejora de la composición *LoanApprovalService* que se describe en la documentación del estándar WS-BPEL 2.0 [17]. La composición *ServicioEnvío* (216 LOC) es una ampliación de la composición abstracta *ShippingService* que se suministra con la documentación del estándar WS-BPEL 2.0 [17]. Por último, la composición *SumaCuadrados* (137 LOC) se ha desarrollado en el seno del grupo de investigación UCASE de la Universidad de Cádiz. Estas composiciones están disponibles en [11] y se han ampliado para poder aplicar un conjunto mayor de operadores de mutación.

En conjunto, las cuatro composiciones utilizadas, permiten aplicar todos los operadores de mutación definidos en este trabajo, como puede verse en la Tabla 5.

Tabla 5. Operadores de mutación utilizados en las composiciones

	CV	P	SE	SC	Total
CFA	16	17	14	21	68
CDE	4	4	2	0	10
CCO	10	4	2	0	16
CDC	12	4	2	0	18
EIU	0	2	2	13	17
EIN	2	2	1	0	5
EAP	0	2	2	13	17
EAN	0	2	2	13	17
Total	44	37	27	60	168

El procedimiento seguido para determinar si los nuevos operadores aportan información significativa al proceso de prueba es el siguiente:

1. Se ha diseñado para cada composición utilizada un conjunto de casos de prueba adecuado, es decir, que mate a todos los mutantes no equivalentes producidos por los operadores de [8].
2. Se han ejecutado los mutantes producidos por los nuevos operadores de cobertura[3] frente a estos casos de prueba.
3. Se ha comprobado qué mutantes mueren y cuáles quedan vivos.
4. En el caso de los mutantes que han quedado vivos, se ha comprobado si son equivalentes, o si pueden morir con casos de prueba adicionales.

La Tabla 6 muestra el número de mutantes que mueren, los que quedan vivos y los que son erróneos[4] tras la ejecución de los mutantes de cobertura producidos por cada composición con conjuntos de casos de prueba adecuados para los mutantes producidos por los operadores de [8].

Tabla 6. Resultado de la ejecución de los mutantes con conjuntos de casos de prueba adecuados

	CV			P			SE			SC		
	Muertos	Vivos	Err.	Muertos	Vivos	Err.	Muertos	Vivos	Err.	Muertos	Vivos	Err.
CFA	9	0	7	16	0	1	14	0	0	20	0	1
CDC	10	2	0	4	0	0	1	1	0	-	-	-
EIU	-	-	-	2	0	0	0	2	0	11	2	0
EIN	2	0	0	2	0	0	1	0	0	-	-	-
EAP	-	-	-	0	2	0	0	2	0	0	13	0
EAN	-	-	-	2	0	0	0	2	0	11	2	0

La Tabla 6 muestra que algunos de los mutantes producidos por los operadores de cobertura quedan vivos. Posteriormente hemos comprobado cuáles de ellos eran equivalentes, resultando que lo eran 23 de los 28 mutantes que quedaron vivos. Los otros 5 mutantes vivos son producidos por los operadores CDC (3 mutantes), EIU (1 mutante) y EAN (1 mutante).

Dos de los mutantes no equivalentes generados por CDC mueren con nuevos casos de prueba. El resto de mutantes vivos no equivalentes no han podido ser matados con nuevos casos de prueba porque provocan un bucle infinito y la actual implementación de nuestra herramienta de mutación no los distingue del original. Esto nos ha permitido detectar este fallo y corregirlo.

Por otro lado, también se han ejecutado los mismos mutantes con conjuntos de casos de prueba no adecuados para los operadores definidos en [8]. En este

[3] A la hora de generar los mutantes, no hemos tenido en cuenta los producidos por CDE y CCO ya que el operador CDC subsume a ambos, esto evita la generación de mutantes repetidos.

[4] Algunos operadores pueden producir mutantes que no se pueden ejecutar, estos son los mutantes erróneos o no válidos.

caso se obtienen muchos más mutantes vivos que pueden ser matados por casos de prueba adicionales.

A partir del estudio experimental realizado podemos concluir que, para las composiciones utilizadas, tres de los nuevos operadores de cobertura implementados (CDC, EIU y EAN) proporcionan información adicional útil para el proceso de prueba.

9. Conclusiones y Trabajo Futuro

En este artículo se ha definido un conjunto de operadores de cobertura para el lenguaje WS-BPEL 2.0 con el objetivo de obtener casos de prueba acordes a los criterios clásicos en un procedimiento de prueba de caja blanca. Además, se ha definido otro conjunto de operadores que, sin ser estrictamente operadores de cobertura, pueden realizar dicha función en algunos contextos determinados.

Por otro lado, se ha realizado una comparativa de los operadores de cobertura de WS-BPEL 2.0 definidos en este artículo con los definidos para los lenguajes estructurados Ada, C y Fortran. En la mayoría de los casos existe una equivalencia entre los operadores de cobertura de WS-BPEL y los de estos lenguajes.

Hay que señalar el hecho de que los operadores definidos para C y Fortran normalmente engloban varias mutaciones, mientras que cada operador definido para WS-BPEL realiza un solo tipo de cambio. Así, por ejemplo, el operador ABS de Fortran realiza las mismas mutaciones que los operadores EAP y EAN conjuntamente. Se trata de una decisión de diseño de los operadores de WS-BPEL, debido a que se considera más útil para el usuario disponer de operadores que realicen un único tipo de cambio sintáctico.

También se ha extraído de esta comparativa que los operadores de cobertura definidos para el lenguaje Ada son aplicables a programas WS-BPEL, ya que son idénticos a los definidos en este trabajo. En el caso de Fortran o C, a pesar de las múltiples diferencias, los operadores también son aplicables a código WS-BPEL, teniendo en cuenta que las funcionalidades que aportan están ya recogidas en otros operadores definidos para WS-BPEL en [8], que no tienen la función específica de cobertura.

Por último, se han realizado unos experimentos para comprobar si el nuevo conjunto de operadores de cobertura para WS-BPEL aporta o no información adicional. Hemos concluido que para las composiciones utilizadas en el estudio, los operadores CDC, EIU y EAN aportan información extra respecto a la del conjunto original. Además, su aplicación nos ha permitido encontrar errores en la herramienta MuBPEL.

En cuanto a las líneas futuras de trabajo, la principal sería realizar nuevos experimentos con un mayor número de composiciones y con otros conjuntos de casos de prueba.

Agradecimientos. Este trabajo ha sido financiado por el proyecto MoDSOA (TIN 2011-27242) del Programa Nacional de I+D+i del Ministerio de Ciencia e Innovación y el proyecto PR2011-004 de la Universidad de Cádiz.

Referencias

1. Agrawal, H., Demillo, R., Hathaway, R., Hsu, W., Hsu, W., Krauser, E., Martin, R., Mathur, A., Spafford, E.: Design of Mutant Operators for the C Programming Language. Tech. Rep. SERC-TR-41-P, Software Engineering Research Center, Department of Computer Science, Purdue University, Indiana (1989)
2. Berglund, A., Boag, S., Chamberlin, D., Fernández, M.F., Kay, M., Robie, J., Siméon, J.: XML Path Language (XPath) 2.0. W3C recommendation, W3C (2010), http://www.w3.org/TR/xpath20
3. Biron, P., Malhotra, A.: XML Schema Part 2: Datatypes. W3C recommendation, W3C (2004), http://www.w3.org/TR/xmlschema-2
4. Boubeta-Puig, J., Medina-Bulo, I., García-Domínguez, A.: Analogies and Differences between Mutation Operators for WS-BPEL 2.0 and Other Languages. In: IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops (ICSTW). pp. 398–407 (marzo 2011)
5. Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., Yergeau, F.: Extensible Markup Language (XML) 1.0. W3C recommendation, W3C (2008), http://www.w3.org/TR/xml
6. Chinnici, R., Moreau, J., Ryman, A., Weerawarana, S.: Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. Tech. rep., W3C (2007), http://www.w3.org/TR/wsdl20
7. Delamaro, M., Maldonado, J.: Proteum - A Tool for the Assessment of Test Adequacy for C Programs. In: Conference on Performability in Computing System. pp. 79–95. East Brunswick, Nueva Yersey (1996)
8. Estero-Botaro, A., Palomo-Lozano, F., Medina-Bulo, I.: Mutation Operators for WS-BPEL 2.0. In: 21th International Conference on Software & Systems Engineering and their Applications. Paris, Francia (2008)
9. Estero-Botaro, A., Palomo-Lozano, F., Medina-Bulo, I.: Quantitative Evaluation of Mutation Operators for WS-BPEL Compositions. In: 3rd International Conference on Software Testing, Verification, and Validation Workshops. pp. 142–150. IEEE Computer Society, Paris, Francia (2010)
10. García-Domínguez, A., Estero-Botaro, A., Medina-Bulo, I., Palomo-Lozano, F.: MuBPEL: Una Herramienta de Mutación Firme para WS-BPEL 2.0. In: Actas de las XVI Jornadas de Ingeniería del Software y Bases de Datos (2012)
11. Grupo de investigación UCASE, Universidad de Cádiz: Repositorio de composiciones WS-BPEL (2012), https://neptuno.uca.es/svn/wsbpel-comp-repo
12. Jia, Y., Harman, M.: An Analysis and Survey of the Development of Mutation Testing. IEEE Transactions on Software Engineering 37(5), 649 –678 (octubre 2011)
13. King, K., Offutt, A.: A FORTRAN Language System for Mutation-based Software Testing. Software – Practice and Experience 21(7), 685–718 (1991)
14. Offutt, A.J., Untch, R.H.: Mutation Testing for the New Century, chap. Mutation 2000: Uniting the Orthogonal, pp. 34–44. Kluwer Academic Publishers (2001)
15. Offutt, A., Voas, J., Payne, J.: Mutation Operators for Ada. Tech. Rep. ISSE-TR-96-09, Department of Information and Software Systems Engineering, George Mason University (1996)
16. Offutt, A., Voas, J.: Subsumption of Condition Coverage Techniques by Mutation Testing. Tech. Rep. ISSE-TR-96-01, Department of Information and Software Systems Engineering, George Mason University (1996)
17. Organization for the Advancement of Structured Information Standards: Web Services Business Process Execution Language 2.0 (2007), http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

Prueba de mutaciones sobre consultas de procesamiento de eventos en aplicaciones en tiempo real

Lorena Gutiérrez Madroñal, Juan José Domínguez Jiménez, e Inmaculada Medina Bulo

Dpto. de Lenguajes y Sistemas Informáticos
Escuela Superior de Ingeniería, Universidad de Cádiz
11002 Cádiz
{lorena.gutierrez, juanjose.dominguez, inmaculada.medina}@uca.es

Abstract. La prueba de mutaciones es una técnica de prueba de software que ha sido usada con éxito en la prueba de lenguajes de programación clásicos. Sin embargo, no se ha empleado en la prueba de aplicaciones en tiempo real que procesen un gran número de flujos de eventos y en las que se realicen consultas de procesos de eventos. Un error mientras se está diseñando la consulta para procesar un flujo de eventos, puede ocasionar un comportamiento anómalo del sistema. En este trabajo, proponemos la prueba de mutaciones para controlar las consultas en aplicaciones en tiempo real realizadas en el lenguaje EPL de procesamiento de eventos. Se presentan y definen los operadores de mutación para EPL, comparándolos con los operadores de mutación del lenguaje SQL. Definimos los criterios necesarios para matar mutantes en EPL. Finalmente, se presenta una arquitectura para la generación automática de dichos mutantes.

Keywords: Lenguaje para el Procesamiento de Eventos, Operadores de mutación para EPL, Aplicaciones en tiempo real, Casos de prueba adecuados

1 Introducción

La prueba de mutaciones es una técnica de prueba que introduce fallos simples en el programa a probar que puede medir la efectividad de un conjunto de casos de prueba para la localización de esos fallos. Los *mutantes* son el resultado de introducir esos fallos, aplicando *operadores de mutación*, en el programa a probar. Cada operador de mutación se corresponde con una categoría de error típico que el desarrollador podría cometer. Los pequeños cambios sintácticos que contienen los mutantes con respecto al programa original deben ser detectados por el conjunto de casos de prueba.

Si un programa contiene la instrucción x = 60 y disponemos de operadores de mutación sobre los operadores relacionales (se cambia un operador relacional por otro), el mutante resultante podría tener como instrucción, por ejemplo, $x \neq 60$. Si un caso de prueba es capaz de diferenciar entre el programa original y el mutante, es decir, sus salidas son diferentes, se dice que el caso de prueba mata al mutante. Si ningún caso de prueba es capaz de diferenciar las salidas de ambos, el mutante sigue vivo.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 369-382, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

Una de las dificultades de aplicar la prueba de mutaciones es la existencia de *mutantes equivalentes*. Éstos tienen el mismo comportamiento que el programa original, es decir, siempre producen la misma salida, por lo que no pueden ser diferenciados del programa original. No deben confundirse estos mutantes con los mutantes que sobreviven porque el conjunto de casos de prueba no es adecuado para detectarlos.

La prueba de mutaciones ha demostrado ser efectiva en diversos lenguajes [1 – 8, 18] para conseguir la calidad de un software implementado. Desafortunadamente, no existe suficiente información sobre la prueba de mutaciones en aplicaciones de tiempo real [22 - 25] que procesen un gran número de flujos de eventos. Éstas tienen procesos de eventos complejos que aceptan como entrada flujos de eventos que luego redirigen a "listeners" que se basan en reglas o en eventos que se activan comparando expresiones de patrones. Estas consultas son parcialmente similares a las consultas tradicionales del lenguaje de programación SQL, pero a diferencia de éstas, proporcionan medios para la expresión de algunas características (por ejemplo, expresión de patrones). Un error en el diseño de este tipo de consultas puede provocar desde un anormal comportamiento del programa a la pérdida de muchas oportunidades de negocio. Además, las consultas, al poder ser generadas dinámicamente, pueden disponer de entradas que no han sido filtradas apropiadamente. Esto hace necesario que se preste atención a las pruebas en este tipo de consultas en aplicaciones de tiempo real.

En este artículo proponemos un enfoque basado en la prueba de mutaciones para evaluar las consultas de aplicaciones en tiempo real realizadas en el lenguaje EPL (Event Processing Language) [9] de procesamiento de eventos. A pesar de que EPL es similar al lenguaje SQL (Structured Query Language), existe un número considerable de diferencias entre ambos. Hemos definido y diseñado un conjunto de operadores de mutación para este lenguaje, así como los criterios que se deben emplear para considerar a un mutante muerto. Los operadores que proponemos introducen fallos en las consultas EPL (expresiones de patrones, ventanas de longitud y tiempo, proceso de eventos por lotes), así como en las cláusulas, operadores y en el uso de la palabra reservada NULL. Además también presentamos una arquitectura para generar de manera automática los mutantes para EPL.

Los operadores propuestos y los criterios para matar los mutantes que se presentan pueden ser usados para controlar la calidad de algunos de los aspectos más comunes de las aplicaciones en tiempo real, como errores o retrasos en la notificación de eventos, e inesperadas e incorrectas secuencias de eventos.

Las siguientes secciones se organizan de la siguiente forma: en la Sección 2 se presenta el lenguaje EPL y las diferencias y similitudes con SQL. En la Sección 3 se definen los operadores de mutación, los criterios para matar los mutantes y algunos ejemplos. La Sección 4 describe el framework para EPL. La Sección 5 presenta los trabajos relacionados y la Sección 6 las conclusiones y el trabajo futuro.

2 Esper, EPL y SQL Database

El motor de Esper ha sido desarrollado para satisfacer los requisitos de las aplicaciones que analizan y reaccionan a los eventos. Algunos ejemplos típicos de este tipo de

aplicaciones son la gestión y automatización de procesos de negocio, las finanzas, la monitorización de aplicaciones y redes y las aplicaciones de sensores de redes. Lo que tienen en común todas estas aplicaciones es que necesitan procesar eventos o mensajes en tiempo real, no en un tiempo cercano al real. Esto significa que a veces se tienen que procesar eventos complejos (CEP) y analizar el flujo de estos eventos. Las consideraciones claves para este tipo de aplicaciones son: el rendimiento, la latencia y la complejidad de la lógica necesaria.

Esper es un motor implementado en Java que permite especificar y ejecutar consultas EPL [9]. Esto permite al programador implementar aplicaciones en tiempo real capaces de enviar y escuchar a la vez continuos flujos de eventos. Como los flujos de eventos están disponibles en la aplicación, el programador tiene la opción de implementar filtros para escuchar eventos específicos de interés, relacionados con eventos del mundo real.

Existen diferencias entre Esper y los sistemas de bases de datos tradicionales. Primero, Esper almacena consultas y las bases de datos almacenan datos. Segundo, mientras que una sentencia de SQL puede ser invocada por una aplicación (dependiendo de funcionalidades específicas), una sentencia EPL se ejecuta de inmediato y de forma continua tras haber sido creada durante la ejecución de la aplicación. Además las consultas EPL también se ejecutan cuando se reciben eventos predefinidos en la aplicación o cuando se dispara el temporizador. Tercero, en Esper, el tiempo es una propiedad, a diferencia de las bases de datos que es un tipo de dato. Luego, eventos, flujos de eventos y consultas EPL, pueden ser consideradas análogas a filas, tablas y consultas SQL, respectivamente.

El lenguaje EPL es un lenguaje similar a SQL con las cláusulas: SELECT, FROM, WHERE, GROUP BY, HAVING y ORDER BY. Las cadenas reemplazan a las tablas como fuente de datos y los eventos reemplazan a las filas como unidad básica de datos [9]. La figura 1 muestra un ejemplo de una consulta EPL que informa de un evento si el precio de compra de un artículo supera los 5€, y el precio medio de todos los artículos comprados supera los 8€ en un plazo de 10 minutos.

```
select * from
pattern [every (Order.price > 5.0 -> avg (Order.price) > 8.0))] where timer:within (10 min)]
```

Fig. 1. Ejemplo de un patron de eventos en EPL.

Esper ofrece dos métodos para procesar los eventos: patrones de eventos y consultas de eventos en cadena. Para el análisis de las aplicaciones CEP (aplicaciones que procesan eventos complejos), las consultas de eventos en cadena de Esper son la mejor opción a usar con estos eventos encadenados. Éstas siguen la sintaxis de EPL y proporcionan: ventanas, agregaciones, uniones y análisis de funciones.

3 Operadores propuestos para EPL y criterios para matar a un mutante en EPL

En esta sección se proponen los operadores de mutación para evaluar la calidad de los flujos de eventos que proporcionan las consultas EPL. Los operadores están diseñados

para introducir fallos en determinadas características: en expresiones de patrones, en ventanas de longitud y tiempo variable, y en la capacidad de procesamiento por lotes (Véanse de 3.2 a 3.5). Como EPL también es vulnerable a los ataques de entradas SQL [26], también se diseñan este tipo de operadores (Véase 3.6). Dado que EPL está basado en SQL, además desarrollamos operadores análogos a los ya existentes para SQL [7, 11, 12] (Véanse de 3.7 a 3.9). Del mismo modo se indican los operadores que generan mutantes equivalentes. Pero antes de presentarlos, destacaremos los criterios para matar los mutantes que se van a generar según estos operadores.

3.1 Criterios para matar a un mutante en EPL

En general, las aplicaciones en tiempo real implementan métodos para manejar los eventos e informan sobre ellos. Asumimos que estos métodos están bien implementados mientras que se aplican nuestros operadores de mutación para evaluar si son adecuados los casos de prueba. Vamos a comparar el número total de los eventos obtenidos por la consulta original (O), y por la consulta mutada (M) para decidir si el mutante está vivo o muerto, véase tabla 1. Un mutante está muerto si el número de eventos obtenidos no es el mismo que el del original, siendo estos eventos obtenidos los mismos para ambas consultas.

Categorías	**Operadores**	**Criterios para matar**
Pattern expression (PEP)	RREP, CEOP, RLOP	Número de eventos entre *O* y *M*
	RGEP, OEDIP,OEDDP	Latencia entre *O* y *M*
Window of length (WOL)	LINC, LDEC	Número de eventos entre *O* y *M*
Window of time (WOT)	TINC, TDEC, TRUN	Latencia entre *O* y *M*
Batch processing of event (BOE)	BATL	Número de eventos entre *O* y *M*
	BATT	Latencia entre *O* y *M*
SQL injection attack (SQIJ)	WCRW, WCNG, WCFD	Número de eventos entre *O* y *M*
EPL clause (EPLC)	EAGR, ESEL, EGRU, EJOI, EORD, ESUB	Número de eventos entre *O* y *M*
	ESIR	Número de eventos entre *O* y *M* Latencia entre *O* y *M*
Operator replacement (ORP)	EBTW, EABS, ELKE, EAOR, EROR, EUOI	Número de eventos entre *O* y *M* Latencia entre *O* y *M*.
Null mutation operator (NOP)	ENLF, ENLI, ENLO	Número de eventos entre *O* y *M*

Table 1. Operadores y criterios para matar los mutantes

Algunos de los operadores (WCNG y WCFD), necesitan un criterio diferente para matar los mutantes. Para ellos, un mutante está muerto si el número de eventos obtenidos por la consulta original y la mutada coincide. Con los operadores que introducen fallos en parámetros de tiempo, comparamos la latencia entre la consulta original y la mutada. La latencia de la ejecución de una consulta se calcula basándose en la diferencia entre el tiempo de suministro de las entradas de eventos necesarios y el momento en el que se activa un método de devolución. Luego el retraso de la notifi-

cación de un evento entre una consulta original y una mutada es un criterio para matar un mutante.

3.2 Operadores de mutación para expresiones de patrones

Las consultas EPL pueden incluir expresiones de patrones que se comparan con uno o muchos flujos de eventos de entrada y dependiendo del tiempo, podrían anidarse. Se proponen 6 operadores (véase tabla2):

1. RREP - Remove repetition in pattern expression: Se elimina el operador "*every*" de una expresión. Este permite una comprobación continua de los patrones, y eliminándolo, conseguimos que solamente se haga una comprobación.
2. CEOP - Change event order in pattern expression: Se cambia el orden de la expresión. El orden de los eventos viene definido con el operador "→". Se intercambian los eventos a ambos lados del operador "→".
3. RLOP - Replace logical operator in pattern expression: Se sustituyen, cada uno de los operadores lógicos "*and*" y "*or*" por el otro.
4. RGEP - Remove guard expression: Se elimina la expresión "*guard*" presente en las condiciones WHERE. Estas controlan el ciclo de vida de las expresiones de patrones. Las consultas mutadas, modifican el ciclo de vida del patrón.
5. OEDIP - Observer expression's time delay increment: Se incrementa en uno el valor del tiempo en el observador del patrón. En "*timer:at*" como en "*timer:interval*".
6. OEDDP - Observer expression's time delay decrement: Igual que el operador OEDIP, pero se decrementa en uno el valor.

Nombre	Consulta original	Consulta mutada
RREP	every (p=Order(price>5))	(p=Order(price>5))
CEOP	(a=Order (price > 5 and symbol= "AAPL") → o = Order (price > 8 and symbol="ORNG"))	(o = Order (price > 8 and symbol ="ORNG") → a=Order (price > 5 and symbol= "AAPL"))
RLOP	Order (price > 5 and symbol="AAPL")	Order (price > 5 or symbol="AAPL")
RGEP	select avg(price) from Order where timer.within (10 sec)	select avg(price) from Order
OEDIP	every timer:interval (20 sec)	every timer:interval (21 sec)
OEDDP	every timer:interval (20 sec)	every timer:interval (19 sec)

Table 2. Ejemplos de aplicación de los operadores en expresiones de patrones.

3.3 Operadores de mutación para ventanas de longitud variable

Se pueden incluir ventanas de longitud variable, que indiquen al motor de búsqueda los N últimos eventos a mantener. Se proponen 2 operadores (véase tabla 3).

1. LINC – Increase data window length by one: Una ventana de longitud indica al motor de búsqueda que solo ha de mantener los N últimos eventos para un flujo. El operador LINC, incrementa la longitud de la ventana en uno.

2. LDEC – Decrease data window length by one: Este operador reduce el valor de la longitud de la ventana en uno.

Nombre	Consulta original	Consulta mutada
LINC	select * fromOrder.win:length(5)	select * from Order.win:length (6)
LDEC	select * from Order.win:length (5)	select * from Order.win:length (4)

Table 3. Ejemplos de aplicación de los operadores para ventanas de longitud variable.

3.4 Operadores de mutación para ventanas de tiempo variable

Las ventanas de tiempo variable de EPL permiten restringir el número de eventos a procesar por consulta. Se proponen tres operadores (véase tabla 4).

1. TINC – Increase time window by one: Una ventana de tiempo es una ventana que varía según el tiempo del sistema. Esto permite al programador restringir el número de eventos a procesar por consulta. El operador TINC, incrementa el tiempo especificado en la ventana en una unidad.
2. TDEC – Decrease time window by one: El operador TDEC decrementa el tiempo de la ventana en una unidad.
3. TRUN – Replace timer unit: El operador reemplaza la unidad de tiempo especificada por otra. El conjunto de unidades de tiempo son: milisegundos, segundos, minutos, horas y días.

Nombre	Consulta original	Consulta mutada
TINC	select avg(price) from Order.win:time (4 sec)	select avg(price) from Order.win:time (5 sec)
TDEC	select avg(price) from Order.win:time (4 sec)	select avg(price) from Order.win:time (3 sec)
TRUN	select avg(price) from Order.win:time (4 min)	select avg(price) from Order.win:time (4 sec)

Table 4. Ejemplos de aplicación de los operadores para ventanas de tiempo variable.

3.5 Operadores de mutación para el procesamiento de eventos por lotes

Las consultas EPL pueden incluir el procesamiento de eventos por lotes basados en el tiempo y la longitud de las ventanas. Se proponen dos operadores (véase tabla 5):

4. BATL – Replace batch processing length with ordinary window length: La función "*win:length_batch*" nos permite obtener los datos de un número específico de eventos. Una vez que se obtienen los valores, el evento que está escuchando, obtiene esta información. El operador reemplaza "*win:length_batch*" por "*win:length*".
5. BATT – Replace batch processing time with ordinary window time: La función "*win:time_batch*" nos permite obtener los datos de un específico periodo de tiem-

po. Una vez que se obtienen los valores, el evento que está escuchando, obtiene esta información. El operador reemplaza "*win:time_batch*" por "*win_time*".

Nombre	Consulta original	Consulta mutada
BATL	select * from Order.win:length_batch(5)	select * from Order.win:length (5)
BATT	select * from Order.win:time_batch(5)	select * from Order.win:time(5)

Table 5. Ejemplos de aplicación de los operadores para el procesamiento de eventos por lotes.

3.6 Operadores de mutación para entradas SQL

Para las consultas EPL que sufran ataques de entradas SQL, proponemos tres operadores (véase tabla 6):

1. WCRW - Remove SQL Where Conditions: Eliminación de condiciones WHERE. Esto nos permite generar entradas maliciosas relacionadas con ataques SQL. El mutante muere si observamos resultados similares entre la consulta original y la mutada (ataque tautológico [10]).
2. WCNG - Negation of Expression in Where Conditions: Negación de la expresión WHERE. La consulta mutada muere si al pasar el caso de prueba, el número de eventos no coincide con el de la consulta original.
3. WCFD - Prepend 'false and' after the where keyword: Antepone "*false and*" en las condiciones WHERE. La consulta mutada muere si al pasar un caso de prueba genera los mismos eventos que la consulta original.

Nombre	Consulta original	Consulta mutada	Salida (O)	Salida (M)
WCRW	select * from Order (symbol='' or 1=1) where timer.within(10 sec)	select * from Order (symbol='' or 1=1)	(1, AAPL, 5), (2, ORNG, 2), (3, ORNG, 6), (4, AAPL, 2)	(1, AAPL, 5), (2, ORNG, 2), (3, ORNG, 6), (4, AAPL, 2)
WCNG	select * from Order (symbol='' or 1=1) where timer.within(10 sec)	select * from Order (not symbol='' or 1=1) where timer.within(10 sec)	(1, AAPL, 5), (2, ORNG, 2), (3, ORNG, 6), (4, AAPL, 2)	Nada
WCFD	select * from Order (symbol='' or 1=1) where timer.within(10 sec)	select * from Order (symbol='' or 1=1) false and where timer.within(10 sec)	(1, AAPL, 5), (2, ORNG, 2), (3, ORNG, 6), (4, AAPL, 2)	(1, AAPL, 5), (2, ORNG, 2), (3, ORNG, 6), (4, AAPL, 2)

Table 6. Ejemplos de aplicación de los operadores de entradas SQL.

3.7 Operadores de mutación para cláusulas EPL

1. EAGR - Agregate functions: Cada función de agregación es reemplazada por otra, tanto si aparecen tras SELECT como en HAVING. Las funciones de agregación en SQL son: *"min", "max", "avg(distinct)", "sum", "sum(distinct)", "count", "count(distinct)"*. Los cambios tienen que tener en cuenta el tipo de los argumen-

tos. Las funciones de agregación en EPL: *"sum", "avg", "count", "max", "min", "median", "stddev", "avedev"*. Según la sintaxis de EPL (1):

Aggregate-function ([all | distinct] expression) (1)

Se incluyen en los cambios anteponer o quitar las palabras claves "*all*", "*distinct*".

2. ESIR - Single-row functions: Cada función single-row es reemplazada por otra (de tipo compatible). Las funciones "single-row": "*case", "cast", "coalesce", "current_timestamp", "exists", "instanceof", "max", "min", "prev", "prior", "prevwindow", "prevcount"* y *"typeof"*. Ejemplos de aplicación de este operador se puede ver en la tabla 7.

Consulta original	**Consulta mutada**
select prev(2, price) from Order retain 10 events	select prior(2, price) from Order retain 10 events
select prevwindow(price) from Trade.win:length(10)	select prevcount(price) from Trade.win:length(10)

Table 7. Ejemplo de aplicación del operador ESIR

3. GRU - Groupings. Eliminación de expresiones GROUP BY: Si la expresión se encuentra en el SELECT o en ORDER BY, tiene que incluirse en una función de agregación para evitar consultas sintácticamente incorrectas. Las funciones de agregación son: *max* y *min*. En el caso de que sólo exista una expresión GROUP BY, la cláusula entera se elimina.
4. EJOI - JOIN clause: Cada palabra clave: "*inner joing", "left outer join", "right outer join", "full outer join", "cross join"*, es reemplazada por otra. En EPL no existe la palabra reservada *"cross join"*.
5. EORD - Ordering of the result set: Las expresiones ORDER BY son eliminadas. Si existe solamente una expresión, se elimina al completo. Cada par de expresiones adjuntas ORDER BY, es intercambiada, al igual que se intercambian las palabras clave "*asc*" y "*desc*". En el caso de no existir una palabra clave, se añade "*desc*".
6. ESEL - Select clause: Intercambia las palabras clave SELECT y SELECT DISTINCT. En EPL además se incluyen, según su sintaxis (2), los intercambios de las palabras claves: *istream* (eventos de inserción), *rstream* (eventos de eliminación) y *irstream* (eventos de inserción y eliminación).

select [istream | irstream | rstream] [distinct]* | expression_list… (2)

Ejemplos de aplicación de este operador se puede ver en la tabla 8. Este operador genera mutantes equivalentes cuando hablamos de la mutación con la palabra clave istream, palabra clave por defecto. Si una consulta no tiene ninguna de las palabras claves istream, rstream e irstream, los eventos son de inserción. Luego los mutantes generados son equivalentes a la consulta original.

Consulta original	Consulta mutada
select rstream * from Order.win:time(30 sec)	select istream * from Order.win:time(30 sec)
select distinct price from Order	select rstream distinct price from Order

Table 8. Ejemplo de aplicación del operador ESEL

7. SUB - Subquery predicates: Considerando un predicado que normalmente utiliza subconsultas como *"e ℜ k(Q)"*, siendo *"e"* el valor de la fila (atributo o expresión), *"ℜ"*representa un operador relacional, *"k"* es la palabra clave que representa el predicado y *"Q"* la subconsulta. Dependiendo de *"k"*, encontramos tres tipos de predicados:

   - Tipo 1 ($k \in$ {all, any, some}): De la forma *e ℜ k(Q)*
   - Tipo 2 ($k \in$ {in, not in}): De la forma *ek(Q)*
   - Tipo 3 ($k \in$ {exists, no exists}): De la forma *k(Q)*

   Cada palabra clave es reemplazada por otra del mismo tipo, evitando cambios equivalentes. Los predicados de tipo 1, son cambiados a predicados de tipo 2 y 3. Y los de tipo 2 son cambiados a predicados de tipo 1 (combinando todos los operadores relacionales) y 3.

3.8 Operadores de mutación de reemplazamiento.

1. EABS - Absolute Value Insertion: Cada expresión aritmética o referencia a un número "*e"*, es reemplazada por "*abs(e)"* y "*-abs(e)"*. No se realizan estos cambios si "*e"* se encuentra en expresiones GROUP BY, ORDER BY o, dentro de subconsultas EXISTS, en SELECT. Este operador genera mutantes equivalentes, ya que tanto si el número *e* es positivo, como negativo, el reemplazo por *abs(e)*, o *–abs(e),* respectivamente, genera un mutante equivalente con respecto a la consulta original.
2. EAOR - Arithmetic operator replacement: Cada operador aritmético *"=", "-", "*", "/", "%"* es reemplazado por otro.
3. EBTW - Between predícate: Cada condición "*a between x and y*" es reemplazado por "$a > x$ *and* $a \leq y$", y por "$a \geq x$ *and* $a < y$". Ejemplos de aplicación de este operador se puede ver en la tabla 9.

Consulta original	Consulta mutada
select * from Order where price between 50 and 60	select * from Order where price > 50 and price <= 60
select * from Order where price between 50 and 60	select * from Order where price >= 50 and price < 60

Table 9. Ejemplo de aplicación del operador EBTW

4. ELKE - Like predicate: Modifica las condiciones en expresiones LIKE. Se eliminan, reemplazan o añaden caracteres especiales: "*%", "_}", "...)"*. Los caracteres especiales en EPL son: "*_"* y "*%"*.

5. EROR - Relational operator replacement: Cada operador relacional *"="*, *"<>"*, *"<"*, *"≤"* *">"*, *"≥"* es reemplazado por otro.
6. EUOI - Unary operator insertion. Cada expresión aritmética o referencia a un número "*e*" es reemplazado por *"-e"*, *"e + 1"* y "*e – 1*".

3.9 Operadores de mutación para la palabra reservada NULL

1. ENLF - Null check predicates: Intercambia los predicados "*is null*" y" *is not null*".
2. ENLI - Include nulls: Establece el valor de la condición a "true" donde hay un valor NULL. Cada atributo *a* en una condición *C* es reemplazado por "*C or a is null*".
3. ENLO - Other nulls. Cada atributo a en condición *C* es reemplazado por: "*not C or a is null"*, *"a is null"* y *"a is not null"*.

4 Arquitectura para la prueba de mutaciones en EPL

Recientemente en el grupo de investigación UCASE1, hemos desarrollado la herramienta MuBPEL [8], para realizar la prueba de mutaciones en el lenguaje Web Services Business Process Execution 2.0. (WSBPEL). MuBPEL se utiliza para evaluar la calidad de un conjunto de casos de prueba, comprobando si el mutante puede ser considerado diferente al programa original.

Los operadores de mutación diseñados anteriormente (sección 3) se probarán en una herramienta que automatice la generación de mutantes en EPL cuyo esquema (Figura 2) es similar al de MuBPEL.

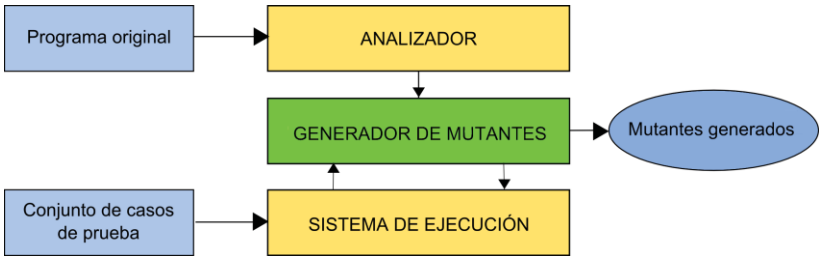


Fig. 2. Arquitectura de la herramienta para la generación de mutantes EPL

Esta arquitectura consta de tres partes: análisis del programa original, generación de mutantes y ejecución de los mutantes. Se recibe como entrada el lenguaje original y un conjunto de casos de prueba. El analizador, toma el programa original y determina los operadores de mutación que se pueden aplicar. Esta información la recibe el segundo componente, el generador de mutantes. Finalmente el sistema de ejecución determinará, con un conjunto de casos de prueba suministrado por el usuario, qué mutantes están vivos o muertos, y con ello podrá comprobar la calidad de los mutantes generados. La calidad de los casos de prueba es evaluada basándose en la puntua-

[1] https://neptuno.uca.es/redmine/

ción de mutación (relación entre el número de mutantes no equivalentes y el número de mutantes muertos). Esta arquitectura acepta dos tipos de casos de prueba: flujos de eventos de datos y entradas que pueden ser parte de consultas dinámicas en EPL. Para la validación de los operadores de mutación de EPL aplicaremos en MuEPL un conjunto de casos de prueba que abarque todos los operadores definidos. Actualmente no existe un conjunto de casos de prueba de consultas EPL para la prueba de mutaciones. A pesar de encontrarse disponibles en la web de Esper consultas EPL [9], éstas utilizan eventos de tipos distintos, siendo inconsistentes y no siendo útiles para un conjunto de casos de prueba. Por esto, y tras estudiar nuestros operadores, se ha tenido que elaborar un nuevo conjunto de casos de prueba que contempla las características del lenguaje EPL2.

5 Trabajos relacionados

Comenzamos esta sección con algunos trabajos relacionados que emplean operadores de mutación para probar aplicaciones en tiempo real detallando más aquellos relacionados con nuestro trabajo. Mark et al [22] generan mutantes utilizando documentos gráficos de aplicaciones en tiempo real para generar casos de prueba adecuados de tal modo que pueden solventar algunos de los problemas de las aplicaciones en tiempo real. Los mutantes se generan en la transición relacionada con los diagramas de estado de modo que las relaciones se realizan con una mayor demora. Nosotros proponemos operadores que aumentan y disminuyen el tiempo de retardo en la ejecución de la consulta. Sin embargo, nuestro objetivo es generar casos de prueba capaces de revelar fallos en los datos y brechas en la seguridad. Geist et al. [23] implementan un conjunto de operadores de mutación para generar casos de prueba de tal modo que las distribuciones de software en tiempo de ejecución se diversifican. Se introducen faltas en las entradas de los datos de los sensores para permitir la diversidad en la ejecución del software en tiempo real. Del mismo modo nosotros diseñamos operadores para modificar las entradas de los usuarios, por lo que las entradas maliciosas y los flujos de eventos diversos pueden revelar fallos en la seguridad y el rendimiento de las aplicaciones en tiempo real. Nilson et al. [24] desarrollan tres operadores de mutación para comprobar el control de los plazos del sistema en tiempo real. Estos permiten un control para implementar casos de prueba por lo que la violación de los plazos puede verse fácilmente. Del mismo modo Sung et al. [25] implementa operadores de mutación para probar la interfaz entre sistemas en tiempo real autómatas. En vez de tanto esfuerzo, nosotros implementamos operadores de mutación para generar casos de prueba, que revelarán cualquiera de estos fallos.

Ahora vamos a hablar sobre algunos trabajos relacionados con las pruebas de mutaciones. Varios trabajos presentan el desarrollo de herramientas para automatizar la generación de mutantes: Mothra [4] para Fortran, MuJava [5] para Java, Proteum [6] para C, SQLMutation [7] para SQL. Su principal característica es que generan au-

[2] https://neptuno.uca.es/svn/sources-fm/trunk/src/muepl/src/test/java/sourcefiles/FilterQueries/epl-benchmark-May2012

tomáticamente todos los posibles mutantes para todos los operadores de mutación proporcionados, sin realizar ninguna mutación selectiva en la generación.

Uno de los principales inconvenientes de la prueba de mutaciones es el coste computacional que supone la ejecución de la gran cantidad de mutantes que se generan a partir del programa original. Para reducir el tiempo de ejecución se han propuesto diversos tipos de técnicas que recopilan Jia y Harman [16] y por Polo y Reales [17]. Algunas de éstas se basan en la reducción del número de mutantes que se generan, así tenemos: muestreo de mutantes [2, 18], agrupamiento de mutantes [3], mutación selectiva [19], mutación de orden superior [2] y mutación evolutiva [20].

En el grupo de investigación UCASE hemos desarrollado la técnica de prueba Mutación Evolutiva (ME) [20], la cual genera y ejecuta solo un subgrupo de todos los mutantes. Los mutantes son seleccionados mediante algoritmos genéticos, sin pérdida de efectividad en las pruebas. En trabajos previos, se ha presentado GAmera [14, 15], un sistema para la generación automática de mutantes para composiciones WS-BPEL. GAmera emplea la ME y puede generar, ejecutar, evaluar y clasificar los mutantes según la calidad de éstos. GAmera está desarrollada de tal forma que puede convertirse en un sistema para la generación automática de mutantes para cualquier lenguaje de programación. Incorpora un algoritmo genético que selecciona sólo los mutantes de mayor calidad, reduciendo el coste computacional que supondría la ejecución de todos los mutantes. Debido a sus características, se quiere adaptar este sistema de generación automática de mutantes a otros lenguajes de programación, como EPL. Con esto se consigue que la técnica de prueba ME se utilice con este nuevo lenguaje, abriendo un nuevo camino para comprobar la efectividad de la misma.

La motivación de nuestro trabajo viene en gran parte del gran número de trabajos que proponen operadores de mutación para las consultas SQL [1, 7, 10 – 15]. Sin embargo, esos trabajos no consideran las características propias de las consultas EPL, tales como las ventanas de eventos tanto de tiempo como de longitud. Los criterios para matar a los mutantes propuestos en estos trabajos no pueden ser aplicados directamente en las consultas EPL. Además, los casos de prueba que proponen, están descritos basándose en las tablas y filas de las bases de datos. Nuestro trabajo propone nuevos operadores y criterios para matar a los mutantes para las consultas EPL. Del mismo modo, se generan casos de prueba específicos para EPL (flujos de eventos y entradas maliciosas), que no se contemplan en los trabajos anteriores.

En [21] se presenta una primera comparación entre los operadores de mutación de los lenguajes (SQL y EPL) y se describen algunos cambios que tienen que sufrir los operadores de SQL para ser adaptados al nuevo lenguaje. Este artículo describe de forma detallada un conjunto de operadores de mutación para EPL, sus criterios para matar a un mutante, junto con la generación de los casos de prueba. Shahriar et al. [10] proponen operadores de mutación que introducen ataques SQL relacionados con los casos de prueba en aplicaciones web. Sin embargo, el trabajo se enfoca en introducir ataques SQL con consultas SQL comunes y con llamadas a eventos que relacionan la API de la base de datos. Este trabajo introduce ataques SQL para consultas EPL en el contexto de aplicaciones en tiempo real.

6 Conclusiones y trabajo futuro

Probar aplicaciones en tiempo real, las cuales consumen y procesan grandes flujos de eventos es un reto. La prueba de mutaciones puede ser una técnica complementaria para evaluar la calidad de las aplicaciones en tiempo real. Los problemas de calidad pueden resolverse realizando pruebas en las consultas EPL de las aplicaciones. Desafortunadamente las investigaciones existentes no están enfocadas en esta dirección.

Este trabajo desarrolla operadores de mutación nóveles y los criterios para matar a un mutante, para evaluar la calidad de las consultas EPL. Los operadores que proponemos pueden ser aplicados para evaluar y generar casos de prueba efectivos, de forma que en los flujos de eventos de datos, se pueden revelar errores de implementación en específicas expresiones de patrones, en ventanas de tiempo y longitud variable y en los eventos procesados por lotes. Desarrollamos operadores para generar entradas dinámicas maliciosas que resultan tras entradas SQL en consultas EPL. También se habla sobre los operadores que son necesarios para probar la exactitud de las cláusulas, operadores y el uso correcto de la palabra reservada NULL en las consultas EPL. Por último, se presenta una arquitectura para la prueba sistemática de consultas EPL.

Nuestro trabajo futuro incluye abarcar diversas direcciones. Se está implementando una aplicación similar a MuBPEL [8] para EPL, la cual se llama MuEPL. MuEPL, analizará, generará todos los mutantes implicados y los ejecutará contra el programa original. Actualmente MuEPL analiza las consultas EPL determinando los operadores de mutación que se pueden aplicar. Tras analizar las consultas, y según los operadores de mutación que se apliquen, se generan los mutantes. MuEPL también podrá ser usada para hacer la evaluación experimental en las aplicaciones en tiempo real que ejecutan consultas EPL, evaluar la calidad de los flujos de eventos y la efectividad de la técnica de prueba ME. Estudiaremos el desarrollo de operadores basados en la calidad específica de aplicaciones en tiempo real, que pueden relacionarse con el rendimiento y la ejecución de las consultas EPL. Un inconveniente de la prueba de mutaciones es la existencia de mutantes equivalentes; también planeamos identificar los operadores que generan este tipo de mutantes para probar las aplicaciones matando a un menor número de mutantes.

7 Agradecimientos

Este trabajo ha sido financiado por el proyecto MoDSOA (TIN2011-27242) del Programa Nacional de Investigación, Desarrollo e Innovación del Ministerio de Ciencia e Innovación y por el proyecto PR2011-004 del Plan de Promoción de la Investigación de la Universidad de Cádiz.

8 Referencias

1. W. K. Chan, S. C. Cheung, T. H. Tse “Fault-based testing of database application programs with conceptual data model”, Proc. 5th Annual International Conference on Quality Software (QSIC 2005) IEEE Computer Society Press, pp. 187 – 198

2. T.A. Budd, Mutation Analysis of Program Test Data. Ph.D. Thesis, Yale University, 1980.
3. S. Hussain, Mutation Clustering. Master's thesis, King's College London 2008.
4. N. King, J. Offutt "A FORTRAN language system for mutation-based software testing" Software - Practice and Experience 21(7), 1991, pp 685 – 718.
5. Y. Ma, J. Offutt, Y. R. Kwon "MuJava: An automated class mutation system" Software Testing, Verification and Reliability 15 (2) 2005, pp 97 - 133.
6. M.E. Delamaro, J.C. Maldonado, "Proteum - a tool for the assessment of test adequacy for C programs", Proc. Conf. on Performability in Computing Systems (PCS 96), pp 79 - 95.
7. J. Tuya, M. Suárez-Cabal, C. de la Riba, "SQLMutation: a Tool to Generate Mutants of SQL Database Queries", 2nd Workshop on Mutation Analysis, 2006, pp 1.
8. MuBPEL website: https://neptuno.uca.es/redmine/projects/ sources-fm/wiki/MuBPEL.
9. EsperTech: http://esper.codehaus.org/.
10. H. Shahriar, M. Zulkernine, "MUSIC: Mutation-based SQL injection vulnerability checking," Proc. of the 8th IEEE International Conf. on Quality Software, pp. 77-86.
11. A. Derezinska "An Experimental Case Study to Applying Mutation Analysis for SQL Queries" Computer Science and Information Technology, 2009.
12. J. Tuya, M. Suárez-Cabal, C. de la Riva, "Mutating database queries", Information and Software Technology, Vol. 49, Issue 4, 2007, pp 398 - 417.
13. H. Shahriar, Mutation-based testing of buffer overflows, SQL injections, and format string bugs, Thesis, Queen's University, 2008 http://qspace.library.queensu.ca/handle/1974/1359.
14. J.J. Domínguez-Jiménez, A. Estero-Botaro, I. Medina-Bulo,"A framework for mutant genetic generation for WS-BPEL", SOFSEM 2009, Proc. of the 35th Conference on Current Trends in Theory and Practice of Computer Science, (5404), Springer-Verlag, pp 229-240.
15. A. Estero Botaro, J. Domínguez Jiménez, L. Gutiérrez Madroñal, I. Medina Bulo "Evaluación de la calidad de los mutantes en la prueba de mutaciones" Proc. XVI JISBD 2011.
16. Y. Jia, M. Harman, "Higher order mutation testing" Information and Software Technology, Vol. 51, Issue 10, 2009, pp. 1379-1393.
17. M. Polo Usaola, P. Reales Mateo, "Mutation testing cost reduction techniques: A survey" IEEE Software, Vol 27, No.3, 2010, pp. 80 - 86.
18. A.T. Acree, On Mutation, Ph.D. Thesis, Georgia Institute of Technology, 1980.
19. A. J. Offutt, A. Lee, G. Rothermel, R. H. Untch, C. Zapf, "An experimental determination of sufficient mutant operators" ACM Transactions on Software Engineering. and Methodology. 5, 1996, pp 99-118.
20. J.J. Domínguez-Jiménez, A. Estero-Botaro, A. García-Domínguez, I. Medina-Bulo, "Evolutionary mutation testing" Information and Software Technology 2011. http://dx.doi.org/10.1016/j.infsof.2011.03.00.
21. L. Gutierrez-Madroñal, J.J. Domínguez-Jiménez, I. Medina-Bulo, "Análisis del lenguage EPL para la prueba de mutaciones", Proc. 3rd JORPRESI, 2011, pp. INF37-INF40.
22. M. Trakhtenbrot, "Implementation-Oriented Mutation Testing of Statechart Models," Proc. 3rd ICSTW, 2010, pp. 120-125.
23. R. Geist, A. Offutt, and F. Harris, "Estimation and Enhancement of Real-Time Software Reliability Through Mutation Analysis," IEEE Transactions on Computers -Special issue on Fault-tolerant Computing, Vol. 41, Issue 5, 1992, pp. 550-558.
24. R. Nilsson and J. Offutt, "Automated Testing of Timeliness: A Case Study," Proc. of the 2nd International Workshop on Automation of Software Test, 2007, pp. 11-18.
25. A. Sung, J. Jang, and B. Choi, "Fault-Based Interface Testing Between Real-Time Operating System and Application," Proc. 2nd ISSRE Workshop Mutation Analysis, 2006, pp. 8.
26. SQL Injection – OWASP, Accessed: https://www.owasp.org/index.php/SQL_Injection

Testing in Service Oriented Architectures with dynamic binding: A mapping study

Marcos Palacios, José García-Fanjul, Javier Tuya

Department of Computing, University of Oviedo
Campus Universitario de Gijón. 33204, Asturias, Spain

{palaciosmarcos, jgfanjul, tuya}@uniovi.es

1 Summary

This article aims at identifying the state of the art in the research on testing Service Oriented Architectures (SOA) with dynamic binding. Testing SOA presents new challenges to researchers because some traditional testing techniques need to be suitably adapted due to the unique features of this new paradigm, for example, the dynamic behavior that allows discovering, selecting and binding a service at runtime. Testing this dynamic binding is one of the most challenging tasks in SOA because the final bound services cannot be known until the moment of the invocations. Hence, there have been a number of recent studies that aim at improving the quality of the dynamic binding using testing approaches.

The objective of this review is to search, analyze and synthesize the different approaches that have been previously proposed. Thus, following the guidelines proposed by Prof. Barbara Kitchenham we have performed a mapping study, which is a particular form of systematic literature review (SLR) that contributes to identify and categorise the available research on a specific topic.

The mapping study has been carried out following a protocol we have developed to guide the search, selection and synthesis of the studies. This protocol includes a set of research questions and a three-phased strategy that allows searching in a broad number of journals and conferences/workshops proceedings. With the aim of selecting the most relevant studies, we have devised study selection criteria that allow deciding whether a study is finally included or excluded in the set of primary studies. Before applying these criteria, we found 392 papers. Removing the duplicates and excluding such papers that do not pass the different criteria, a set of 33 primary studies were finally selected.

As a result of this review, the objectives of the different approaches are grouped into two categories: studies that aim to detect faults in the service oriented application (19 studies) or studies that make a decision about the service to be invoked based on the test results (14 studies). The proposed testing approaches focus more on non-functional characteristics rather than on functional.

Regarding the applied testing techniques, the results of this review show that, currently, two thirds of the studies apply monitoring approaches to improve the dynamic binding. These approaches check properties of the executing system in order to perform an adaptive action (for example, rebind to another service) when a

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 383-384, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

deviation from the expected behaviour is detected. In addition, a third of studies generates and executes test cases with the aim of detecting problems or gathering data to make a decision about the binding.

Although there are different stakeholders that participate in the testing process (provider, registry, client and even a third party), it is the client who plays the most active role with 24 of the 33 studies proposing that the client takes part in the tests. The registry and a broker also represent stakeholders that are often proposed to take part in the tests. However, only four studies suggested that the service providers should participate in the dynamic binding testing process.

During the review we have identified four points in time when the execution of tests may improve the dynamic binding of the services. Service execution is the most frequently recommended point in time to perform tests using monitoring techniques. The points in time before the publication of the services in a registry, during the time they are published and just before their binding are also considered as suitable times to test services.

Regarding the technologies that are used in the primary studies, the description of the atomic services and service compositions that represent the system under test is almost always specified using WSDL for the former and BPEL for the latter, although there are a reduced number of examples that use semantic technologies such as OWL-S or SAWSDL. In such studies where a registry is in charge of storing the services, the UDDI standard is the only mentioned specification. However, in the context of testing SOA with dynamic binding, there is no standard language for representing the terms agreed in an SLA.

The validation methods used in the primary studies have several limitations. Firstly, almost a third of these studies do not present any type of validation of the proposed approach. In addition, the most frequently used validation method is evaluation, with very few studies performing a rigorous analysis of their results and only two studies applying the approach to a real scenario. Most of the examples used in the studies were designed ad hoc and in only one case the example has been extracted from a standard specification.

Reference

1. Palacios, M., García-Fanjul, J., Tuya, J. Testing in Service Oriented Architectures with dynamic binding; a mapping study. Information and Software Technology, 53 (3), 171-189 (2011) http://dx.doi.org/10.1016/j.infsof.2010.11.014

* This work was partially funded by the Department of Science and Innovation (Spain) and ERDF funds within the National Program for Research, Development and Innovation, Projects Test4SOA (TIN2007-67843-C06-01) and Test4DBS (TIN2010-20057 -C03-01) and FICYT (Government of the Principality of Asturias) Grant BP09-075.

Automated Metamorphic Testing on the Analysis of Feature Models *

Sergio Segura[1], Robert M. Hierons[2], David Benavides[1] and Antonio Ruiz-Cortés[1]

[1] University of Seville. Av Reina Mercedes S/N, 41012 Seville, Spain,
[2] Brunel University. Uxbridge, Middlesex, UB7 7NU United Kingdom

1 Summary

Software Product Line (SPL) engineering is a reuse strategy to develop families of related systems. From common assets, different software products are assembled reducing production costs and time–to–market. Products in SPLs are defined in terms of features. A *feature* is an increment in product functionality. *Feature models* are widely used to represent all the valid combinations of features (i.e. products) of an SPL in a single model in terms of features and relations among them. The automated analysis of feature models deals with the computer–aided extraction of information from feature models. Typical operations of analysis allow determining whether a feature model is void (i.e. it represents no products), whether it contains errors (e.g. features that cannot be part of any product) or what is the number of products of the SPL represented by the model. Catalogues with up to 30 analysis operations on feature models and multiple analysis solutions have been reported.

Feature model analysis tools deal with complex data structures and algorithms. This makes the implementation of analyses far from trivial and easily leads to errors increasing development time and reducing reliability of analysis solutions. Gaining confidence in the absence of faults in these tools is especially relevant since the information extracted from feature models is used all along the SPL development process to support both marketing and technical decisions. Thus, the lack of specific testing mechanisms in this context appears as a major obstacle for engineers when trying to assess the functionality and quality of their programs.

In [1], we gave a first step to address the problem of functional testing on the analyses of feature models. In particular, we presented a set of manually designed test cases, the so-called FaMa Test Suite (FaMa TeS), to validate the implementation of the analyses on feature models. Although effective, we found several limitations in our manual approach that motivated this work. First, evaluation results with artificial and real faults showed room for improvement in terms of efficacy. Second, the manual design of new test cases relied on the ability of

* This work has been partially supported by the European Commission (FEDER) and Spanish Government under CICYT project SETI (TIN2009-07366), and by the Andalusian Government under ISABEL(TIC-2533) and THEOS (TIC-5906) projects.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 385-386, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

the tester to decide whether the output of an analysis was correct. We found this was time–consuming, error–prone and in most cases infeasible due to the combinatorial complexity of the analyses. As a result, we were force to use small and in most cases oversimplistic input models whose output could be calculated by hand. This limitation, also found in many other software testing domains, is known as the *oracle problem* i.e. impossibility to determine the correctness of a test output.

Metamorphic testing was proposed as a way to address the oracle problem. The idea behind this technique is to generate new test cases based on existing test data. The expected output of the new test cases can be checked by using known relations (so–called *metamorphic relations*) among two or more input data and their expected outputs. Key benefits of this technique are that it does not require an oracle and it can be highly automated.

In [2], we propose using metamorphic testing for the automated generation of test data for the analyses of feature models. In particular, we present a set of metamorphic relations between feature models and their set of products and a test data generator based on them. Given a feature model and its known set of products, our tool generates a set of neighbouring models together with their associated sets of products. Complex feature models representing millions of products can be efficiently generated by applying this process iteratively. Once generated, products are automatically inspected to get the expected output of a number of analyses over the models. Key benefits of our approach are that it removes the oracle problem and is highly generic being suitable to test any operation extracting information from the set of products of a feature model. In order to show the feasibility of our approach, we evaluated the ability of our test data generator to detect faults in three main scenarios. First, we introduced hundreds of artificial faults (i.e. mutants) into three of the analysis components integrated into the FaMa framework and checked the effectiveness of our generator to detect them. As a result, our automated test data generator found more than 98.5% of the faults in the three reasoners with average detection times under 7.5 seconds. Second, we developed a mock tool including a motivating fault found in the literature and checked the ability of our approach to detect it automatically. As a result, the fault was detected in all the operations tested with a score of 91.4% and an average detection time of 23.5 seconds. Finally, we evaluated our approach with recent releases of two real tools for the analysis of feature models, FaMa and SPLOT, detecting two defects in each of them.

References

1. S. Segura, D. Benavides, and A. Ruiz-Cortés. Functional testing of feature model analysis tools: a test suite. *Software, IET*, 5(1):70 –82, 2011.
2. S. Segura, R.M. Hierons, D. Benavides, and A. Ruiz-Cortés. Automated metamorphic testing on the analyses of feature models. *Information and Software Technology*, 53(3):245 – 258, 2011.

Automated testing on the analysis of variability-intensive artifacts: An exploratory study with SAT Solvers *

Ana B. Sánchez and Sergio Segura

Department of Computer Languages and Systems, University of Seville,
Av Reina Mercedes S/N Seville, Spain

Abstract. The automated detection of faults on variability analysis tools is a challenging task often infeasible due to the combinatorial complexity of the analyses. In previous works, we successfully automated the generation of test data for feature model analysis tools using metamorphic testing. The positive results obtained have encouraged us to explore the applicability of this technique for the efficient detection of faults in other variability-intensive domains. In this paper, we present an automated test data generator for SAT solvers that enables the generation of random propositional formulas (inputs) and their solutions (expected output). In order to show the feasibility of our approach, we introduced 100 artificial faults (i.e. mutants) in an open source SAT solver and compared the ability of our generator and three related benchmarks to detect them. Our results are promising and encourage us to generalize the technique, which could be potentially applicable to any tool dealing with variability such as Eclipse repositories or Maven dependencies analyzers.

1 Introduction

Variability models are a key asset to represent the common and variable features of a configurable software system. The automated analysis of variability models deals with the automated extraction of information from the models [1], e.g. determining the number of possible configurations of a software product. The analysis operations that can be performed on variability models are often very complex [1, 4]. This hinders the testing of these applications making it difficult, often infeasible, to determine the correctness of the outputs, i.e. oracle problem.

Feature models are de-facto standard to manage variability in software product lines [1]. In previous works [4], we presented an automated test data generator for the analysis of feature models overcoming the oracle problem using metamorphic testing [5]. Roughly speaking, we proposed a set metamorphic relations (so-called metamorphic relations) between feature models (inputs) and the set of products that they represent (expected output). Using these relations,

* This work has been partially supported by the European Commission (FEDER) and Spanish Government under CICYT Project SETI (TIN2009-07366), and Projects ISABEL (TIC-2533) and THEOS (TIC-5906) funded by Andalusian Government.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 387-392, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

we enabled the automated generation of random feature models representing up to millions of products that were used as test data. Our results were promising showing the effectiveness of our generator to detect most faults in a few seconds without the need for a human oracle. We also detected two defects in FaMa and another two in SPLOT, two popular open source feature model analysis tools.

The positive results obtained in our previous works have encouraged us to explore the possibility of generalizing our approach to other variability-intensive domains. As a proof of concept, in this paper we propose using metamorphic testing for the automated generation of test data for the analysis of propositional formulas, i.e. SAT problems. In particular, we present a set of metamorphic relations between propositional formulas (input) and their set of solutions (output) and a test data generator based on them. The generator starts by creating an empty formula that is then extended by adding random variables to it. At each step, the set of solutions of the formula is computed by using our metamorphic relations. Complex formulas representing thousands of solutions can be efficiently generated by applying this process iteratively. Once generated, solutions are automatically inspected to get the expected output of the analyses over the formulas, e.g. number of solutions of the formula. In order to show the feasibility of our approach, we introduced 100 artificial faults (i.e. mutants) in a popular open source SAT solver and compared the ability of our generator and three related benchmarks to detect them. As a result, our generator found 98.7% of the faults while the rest of benchmarks only detected 76.3% of them as a maximum. We may remark that although a number of benchmarks for testing SAT solver exists, they focus on the evaluation of the performance and not on the functionality. To the best of our knowledge, this is the first work applying metamorphic testing addressing the automated detection of faults in SAT solvers.

We are aware of only one related work that also addresses the automated detection of faults [2]. This work [2] focused on generating random instances in order to find defects in current solvers. However, in this paper, we focus on generating not only random instances (test data inputs) for SAT solvers but we also obtain expected outputs for these data inputs using metamorphic testing.

2 Propositional formulas

A *propositional formula* consists of a set of literals or variables and a set of logical connectives constraining the values of the variables, e.g. $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$. A SAT solver is a software package that takes as input a propositional formula and determines if the formula is satisfiable, i.e. there is a variable assignment that makes the formula evaluate to true [1]. Input formulas are usually specified in conjunctive normal form (CNF), that is a standard form to represent propositional formulas where only three connectives are allowed: $\neg$, $\wedge$, $\vee$. Propositional formulas in conjunctive normal form consists of the conjunction of a number of *clauses*, where a clause is a disjunction of a number of *literals* or their negations. Consider the following CNF formula: $(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_3)$ where x_i rep-

resents literals and $\neg x_i$ their negations, $\vee$ represents the *or* connective and $\wedge$ represents *and* connective, resulting in a formula with two disjunctions and one conjunction. A possible solution for this formula is x_1=1, x_2=0, x_3=1. Hence, this propositional formula is satisfiable. SAT solving is one of the well known NP-complete problems [3].

3 Automated metamorphic testing on the analyses of propositional formulas

3.1 Metamorphic relations on propositional formulas

In this section, we define a set of metamorphic relations between propositional formulas (i.e. input) and their corresponding set of solutions (i.e. output). Given a propositional formula f in disjunctive format (i.e. a clause), we say that f' is a neighbouring formula if it can be derived from f by adding or removing a literal. Next, we present the metamorphic relations among the solutions of a propositional formula f and the one of their neighbours f':

#R1 -New positive literal-: Consider the formulas and associated set of solutions depicted in Figure 1. f' is created from f by adding a new positive literal (C) to the clause. The relation between the solutions of f and f' can be informally described as follows: the solutions of f' include the solutions of f duplicated and extended with all the possible values of the new literal (i.e. C=1 and C=0). Also, f' adds a new solution with the new literal equal to one and the rest of literals negated.

A	B	f = A v B
0	0	0
0	1	1
1	0	1
1	1	1

Neighbours

Metamorphic relation

A	B	C	f' = A v B v C
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Fig. 1. Neighbour formula. New positive literal.

#R2 -New negative literal-: Consider the formulas and associated set of solutions depicted in Figure 2. f' is created from f by adding a new negated literal ($\neg C$) to the clause. In this case, the solutions of f' include the solutions of f duplicated and extended with all the possible values of the new literal (i.e. C=1 and C=0). Also, f' adds a new solution with the new literal equal to zero and the rest of literals negated.

#R3 -Existing variable with different state-: Consider the formulas $f = A \vee B$ and $f' = A \vee B \vee \neg A$. f' is created from f by adding a literal ($\neg A$) that already existed in the clause with opposite state (A). This is known as tautology, i.e. the formula is true for any variable assignment. This is, the solutions of f' should include all possible literal assignments.

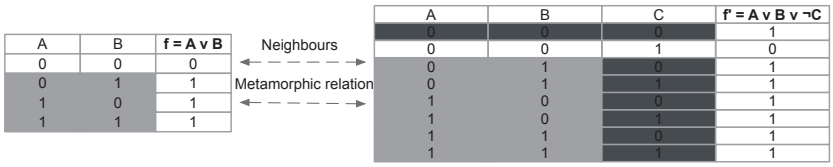


Fig. 2. Neighbour formula. New negative literal.

For completeness, we also considered the case in which an existing variable is added to a clause (e.g. $f = A \vee B$ and $f' = A \vee B \vee A$). In this case, the set of solutions of f' and f remains equal.

3.2 Automated test data generation

We propose a process to automatically generate test data for the analyses of propositional formulas using previous metamorphic relations. The figure 3 illustrates an example of our approach. The generation process is iterative. On each iteration, a clause (i.e. disjunction) and its set of solutions is computed. The generation of each clause starts from an empty formula in which random literals are added creating neighbouring formulas and its corresponding set of solutions according to the metamorphic relations. These iterations end when all the clauses of the formula together with their set of solutions have been generated. At the end of the process, all the clauses are joined using conjunctions and creating a valid CNF formula. The final set of solutions is calculated as the intersection of the set of solutions of all the generated clauses. In the example, a formula with two clauses is created ($f = (A \vee B \vee \neg C) \wedge (A \vee B)$). The formula has five different solutions, i.e. it is satisfiable.

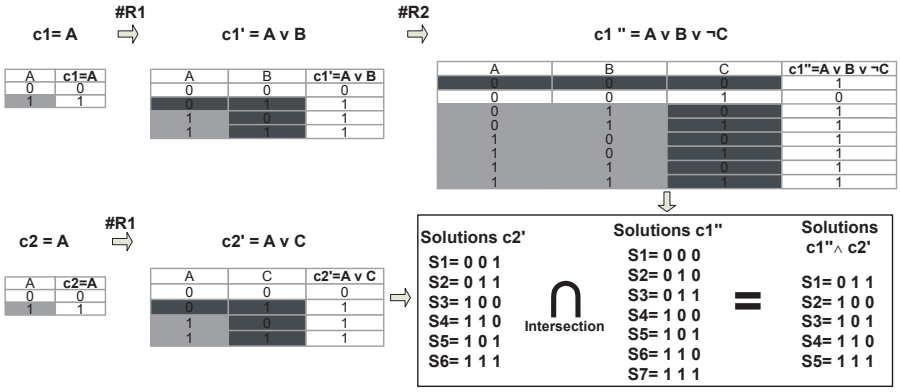


Fig. 3. Example random propositional formula generation using metamorphic relations

4 Preliminary evaluation

As a part of our proposal, we implemented a prototype test data generator. Our tool generates random CNF formulas and its expected solutions. The parameters for the generation are received as input and include, among others, the number of literals, the number of clauses and the number of literals per clause. In order to measure the effectiveness of our proposal, we evaluated the ability of our test data generator to detect faults in the software under test. We applied mutation testing on an open source solver for the analysis of propositional formulas. *Mutation testing* is a common fault-based testing technique that measures the effectiveness of test cases. First, simple faults are introduced into a program creating a collection of faulty versions, called *mutants*. These mutants are created from the original program by applying syntactic changes to its source code. Then, test cases are used to check if the mutants and the original program return different responses. If a test case distinguishes the original program from a mutant we say the mutant has been *killed* and the test case has proved to be effective at finding faults in the program. Otherwise, the mutant remains *alive*. Mutants that keep the program's semantics unchanged and thus cannot be detected are referred to as *equivalents*. *Mutation score* is the percentage of killed mutants with respect to the total number of them (discarding equivalent mutants) and it provides an adequacy measurement of the test suite.

We selected Sat4j[1], a popular open source Java SAT solver as good candidate to be mutated. In particular, we mutated the class *Solver.java* (53 methods, 2,223 lines of code), main class of the solver. To automate the mutation process, we used the mutation tool PIT[2] which showed to be suitable for our approach. Test cases were generated randomly using our prototype tool as described in section 3.1. For the evaluation of our approach, we followed three steps: First, we checked whether the original Sat4j solver passed all the test before its analyses. All the test cases were passed successfully. Second, we generated the mutants by applying all the mutation operators available in PIT, a total of 12. Finally, for each mutant, we ran our test data generator and tried to find a test case that kills it (a maximum of 250 test cases were generated and ran for each mutant).

Table 1 shows the evaluation results after comparing our approach with other 3 benchmarks from the library SATLIB[3]. Out of the 100 generated mutants, 22 were manually discarded because they modified code not covered by any of the benchmarks under evaluation nor our tool. We found that most of them affected secondary functionality of the program not addressed by the tests (e.g. computation of statistics). Also, we discarded two more mutants not exercised by any of the benchmarks and whose semantic equivalence was not clear. This resulted in a total of 76 mutants being evaluated. Results revealed that our generator found 98.7% of the faults (i.e. 75 out of 76) while the rest of benchmarks only detected 76.3% of them as a maximum. We may mention that our generator provides

[1] `http://www.sat4j.org/`
[2] `http://pitest.org/`
[3] `http://www.satlib.org/`

information about the satisfiability of the formula and its exact set of solutions, while the benchmarks only report about the satisfiability of the problem. Hence, with our generator it is possible a more rigorous comparison of the results. For this reason, our generator got a higher score even with simpler formulas. Finally, we generated 250 test cases for efficiency but results suggest that the mutation score could reach 100% by increasing the number of test cases generated.

Test data	Literals	Clauses	Formulas	Mutants	Alive	Score
Our generator	12	24	250	76	1	98.7
SATLIB uf20-91	20	91	250	76	38	50.0
SATLIB uf50-218	50	218	250	76	27	64.5
SATLIB uf20-91-2	20	91	250	76	18	76.3

Table 1. Evaluation results

5 Conclusions and future work

In this paper, we presented a set of metamorphic relations between propositional formulas and a test data generator based on them. Our results show that the application of metamorphic testing in the domain of SAT solvers is effective in detecting most faults without need for a human oracle. To the best of our knowledge, this is the first work applying metamorphic testing addressing the automated detection of faults in SAT solvers.

We identify several challenges for our future work. First, we intend to work in the formal definition of the metamorphic relations and in a more exhaustive evaluation of our test data generator. Secondly, our results are promising and encourage us to generalize this technique. SAT problems are the base of many variability analysis tools such as web configurators, Eclipse repositories or Debian packages repositories analyzers. In fact, this work has allowed us to automate testing on CUDF documents where we have already been able to test p2CUDF automatically, a tool based on Sat4j. Thus, we envision that the concepts shown here could be applicable to many other variability-intensive domains.

References

1. David Benavides, Sergio Segura, and Antonio Ruiz-Cortés. Automated analyses of feature models 20 years later: A literature review. *Information Systems*, 2010.
2. R. Brummayer, F. Lonsing, and A. Biere. Automated testing and debugging of sat and qbf solvers. In *Conf. on Theory and Applications of Satisfiability Testing*, 2010.
3. S. Cook. The complexity of theorem-proving procedures. In *Conference Record of Third Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.
4. S.Segura, R.Hierons, D.Benavides, and A.Ruiz-Cortés. Automated metamorphic testing on the analyses of feature models. *Information Software Technology*, 2011.
5. Z.Q.Zhou, DH.Huang, TH.Tse, Z.Yang, H.Huang, and TY.Chen. Metamorphic testing and its applications. In *Symposium on Future Software Technology*, 2004.

PrMO: An Ontology of Process-reference Models

César Pardo[1,2], Félix García[2], Francisco J. Pino[1,2], Mario Piattini[2] and María Teresa Baldassarre[2]

[1] IDIS Research Group
Electronic and Telecommunications Engineering Faculty
University of Cauca, Street 5 # 4 - 70.
Kybele Consulting Colombia (Spinoff)
Popayán, Cauca, Colombia.
{cpardo, fjpino}@unicauca.edu.co

[2] ALARCOS Research Group
ITSI, Information Systems and Technologies Department
University of Castilla–La Mancha, Paseo de la Universidad 4, Ciudad Real, Spain
{Felix.Garcia, Mario.Piattini}@uclm.es

[3] Department of Informatics, University of Bari.
SER&Practices, SPINOFF, Via E. Orabona 4, 70126, Bari, Italy
baldassarre@di.uniba.it

Resumen. For a couple of decades, process quality has been considered as one of main factors in the delivery of high quality products. Multiple models and standards have emerged as a solution to this issue, but the harmonization of several models in a company for the fulfillment of its quality requirements is no easy task. The difficulty lies in the lack of specific guidelines and in there not being any homogeneous representation which makes this labor less intense. To address that situation, this paper presents an Ontology of Process-reference Models, called PrMO. It defines a Common Structure of Process Elements (CSPE) as a means to support the harmonization of structural differences of multiple reference models, through homogenization of their process structures. PrMO has been validated through instantiation of the information contained in different models, such as CMMI-(ACQ, DEV), ISO (9001, 27001, 27002, 20000-2), ITIL, Cobit, Risk IT, Val IT, BASEL II, amongst others. Both the common structure and the homogenization method are presented, along with an application example. A WEB tool to support the homogenization of models is also described, along with other uses which illustrate the advantages of PrMO. The proposed ontology could be extremely useful for organizations and other consultants that plan to carry out the harmonization of multiple models.

Keywords: Harmonization of multiple models and standards; homogenization; mapping; integration; ontology, processes.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 393-406, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

1 Introduction

Aiming to provide solutions that allow us to define suitable processes for addressing different needs, a wide range of models and standards have been developed (hereafter called *reference models*), which can be used as process reference models; e.g. ISO/IEC 20000-2, ISO/IEC 27001, ISO/IEC 9001, ITIL, SWEBOK, Cobit, ISO/IEC 12207, CMMI, and so forth. Besides these models, there are different assessment models, such as SCAMPI, ISO/IEC 15504-5, CBA-IPI, EIA/IS 731.2 Appraisal Method, SCE V3.0 Method Description, amongst others.

This mass of models and standards that has emerged means that software organizations can assess and institutionalize new or improved process, becoming more competitive and producing high quality products. They can likewise choose a particular model to cover a specific issue, or select several models to address different needs. Currently, there are different factors that may persuade an organization to consider the need to work with more than one model [1], e.g. (i) market niches with specific models, (ii) improvement of practices from legacy process models, (iii) business positioning, (iv) leveraged or merger corporate (v) systematic search of the capability of the processes, (vi) business growth, among others.

Software organizations have found it difficult to work with more than one model at the same time, however, and they often make a great effort to interpret them, due to the fact that each has been defined from different opinions, work groups, (cultural and political), interests and bodies. Each model therefore uses its own view on quality. That is, each of them defines its own process element structure, scope, orientation, purpose, and other characteristics. This has brought about some problems in the use of the reference models e.g. formal description of process models, compatibility and transformability, and benchmark of process attributes [2].

Bearing all the above in mind, our work has the objective and scope of giving a solution to the problem we have set out, defining an ontology which would be useful for harmonizing the process elements which have been described by different models. Our ontology identifies and makes use of the process elements which it would be made up of and that are also common to any reference model. It can thus be used independently of the reference model to be harmonized. Using the ontology, a common schema or *Common Structure of Process Elements* (CSPE) has been defined. This has allowed the homogenization of the process elements of some models, resolving their differences before carrying out any comparison, mapping, integration or unification. A prototype of tool which makes use of the models' information, homogenized through CSPE, is also shown.

This paper is organized as follows. After this introduction, Section 2 presents an analysis of the related works. Then Section 3 introduces PrMO, the Ontology of Process-reference Models. Section 4 shows the use of the PrMO, as well as a *Common Structure of Process Elements* to support the homogenization of multiple models and a homogenization method to support their application. Section 5 sets out the application of the common structure and homogenization of some process elements of ISO 20000-2, together with an overview of a supporting tool. Finally, conclusions and future work are presented in Section 6.

2 Background

In our search in the literature presented in [1], we have identified a few efforts related to harmonization of multiple models, such as PrIME project of the SEI [3], Enterprise SPICE [4], IT Governance Institute (ITGI) and Office of Government Commerce (OGC) carry out the alignment of Cobit 4.1, ITIL V3 and ISO/IEC 27002 for Business Benefit [5], among other publications and works analyzed. Few of them, however, have proposed solutions to resolve the problems and structural differences arising between models that are being harmonized. Most of them carry out the mappings in a unilateral direction and thereby the process structure of basis model is used as a main structure, e.g. the well known mappings of ISO to CMMI performed by Paulk [6], and Mutafelija & Stromber [7]. However, this solution is good only if the objective is focused from the beginning on the instantiation of the good practices of the base model; this is a situation that is impossible to replicate when the needs of the organizations are different. This point, therefore, makes us aware that the integration of models should be treated differently if we need to harmonize other models, e.g. ITIL and Cobit, or BASEL II and Val IT in the case of banking models, and so on.

Some studies have focused mainly on the development of ontologies to represent the key elements of particular domains; e.g. ontologies for representing the ISO and CMMI models, e.g. CMMI-SW [8], CMMI and ISO/IEC 15504 [9, 10], ISO 9001 and CMMI-SW [11]. Malzahn [12] defines an ontology to link the similarities between several models. Mendes & Abran present the engineering domain ontology developed taking SWEBOK as the basis [13], among others. These ontologies have been defined mainly aiming to understand the structure of the process-based quality approaches. We should add that we have also found other studies focusing on development ontologies for supporting business process integration, but this is a subject that is beyond the scope of this article.

Taking into account the situation set out above, we found most studies have focused mainly on the development of ontologies to represent and/or support the key elements of particular domains. This being so, we did not find any proposal standard that was independent and designed exclusively to support the homogenization of structural differences between multiple reference models before they are compared and/or integrated. Likewise, in contrast to related works analyzed, our proposal intends to provide a more fine-grained level.

3 PrMO: An Ontology of Process-Reference Models

PrMO is a subontology which extends one concept of H2mO [14], *quality model*. H2mO provides a formal and clear definition of the most widely-used techniques, methods and related terms in the harmonization of multiple models. PrMO complements H2mO, by means of establishing and clarifying the key process elements to support the harmonization of multiple models through homogenization of their process structures. In this section, we present an overview of the process architecture ontology designed. We then give a general overview of this and its

instantiation from information contained in different models, such as CMMI-ACQ V1.2, ISO 9001, amongst others. We also provide the definition of a Common Structure of Process Elements (CSPE) and its application in the homogenization of *Specific Goal* (SG) 1 of Agreement Management of CMMI-ACQ. An example of instance of CMMI based on ontology is also shown.

3.1 Concepts of PrMO

The generic process constructors of PrMO have been designed by taking some process elements defined in the process structure of SPEM 2.0 [15], e.g. task and product. Using these standard elements and not others, e.g. process elements of a particular model such as CMMI, ITIL, etc., a homogeneous deal is ensured, which is independent of the process structure of reference models used during their harmonization.

Along with process elements taken from SPEM 2.0, we noted from our experience that it was necessary to add other process elements to give support to the homogenization of the process elements of other models with a higher degree of granularity or level of abstraction. Some examples we could point to and which are not described in detail in SPEM are the process elements of resource, tool and process category. The process elements added were identified from an analysis of a literature review of the commonly-identified process elements which are most widely modeled, presented in [16-23]. This allows us to specify some already-existing process elements more clearly and decompose them better.

Other auxiliary elements have been added too, such as associate elements or decomposed elements from other elements; for example- steps of tasks, in-out artifacts, human resources, time, and so on. Decomposition of elements allows support of the homogenization of process elements of models with a higher degree of detail, e.g. MoProSoft, Cobit 4.1, amongst others. We should add that, given that some concepts have already been defined by other studies, we have taken some concepts such as *Quality Model* and *Measure* of other (sub)ontologies; these are *Software Measure Ontology* and *Measurement Ontology*, which are part of Software Measurement Ontology (SMO) presented in [24]. These sub-ontologies establish and clarify the key elements in the definition of software measures, as well as the terminology related to the act of measuring software.

Taking into account the Representation Formalism for Software Engineering Ontologies known as REFSENO [25], it was possible to establish a basic cluster of *concepts (classes)*, *terminal concept attributes (attributes)* and *nonterminal concept attributes (relationships)* for representing any reference model. We used Protégé-OWL [26] as the tool for the creation of our ontology. Table 1 shows the glossary of the concepts of PrMO, according to REFSENO formalism; due to limits on space we have omitted the description of the terminal and nonterminal concept attributes. In an effort to support the homogenization of different models and the software engineering, some descriptions have been adjusted. A graphical representation of PrMO, both concepts and relationships, is shown in Figure 1, using the UML (Unified Modeling Language).

Table 1. Glossary of concepts in the PrMO.

Concept	Super-concept	Descriptions
Process Category	Concept	A *Process Category* is comprised of interrelated processes. [*New concept*].
Process	Concept	Coherent set of policies, organizational structures, technologies, procedures, purposes, objectives and work products that are needed to design, develop, deploy and maintain a software product. [*Adapted from* [18]].
Activity	Concept	This comprises a set of tasks or actions used to produce and maintain devices and achieve the objectives of the process. The activity includes the procedures, standards, policies and objectives to create and modify a set of work products. [*Adapted from* [16]].
Task	Concept	It is a process element that defines the work done by roles. A task is associated with the input and output products [*Adapted from* [15]].
Product	Concept	The set of artifacts to be developed, delivered and maintained in a project is called the product. The products can be input or output, whether mandatory or optional. Products are in most cases tangible artifacts consumed, produced, or modified by Tasks. [*Adapted from* [29] [15]]
Role	Resource	Describes a set or group of responsibilities, duties and skills required to perform a specific activity. [Adapted from [30]].
Resource	Concept	A resource is an asset that a business needs to have. In the field of software engineering, there are two main resources of importance: the developers and the tools. [Adapted from [17]].
Tool	Resource	The tools automate the execution of certain activities. [Adapted from [16]].

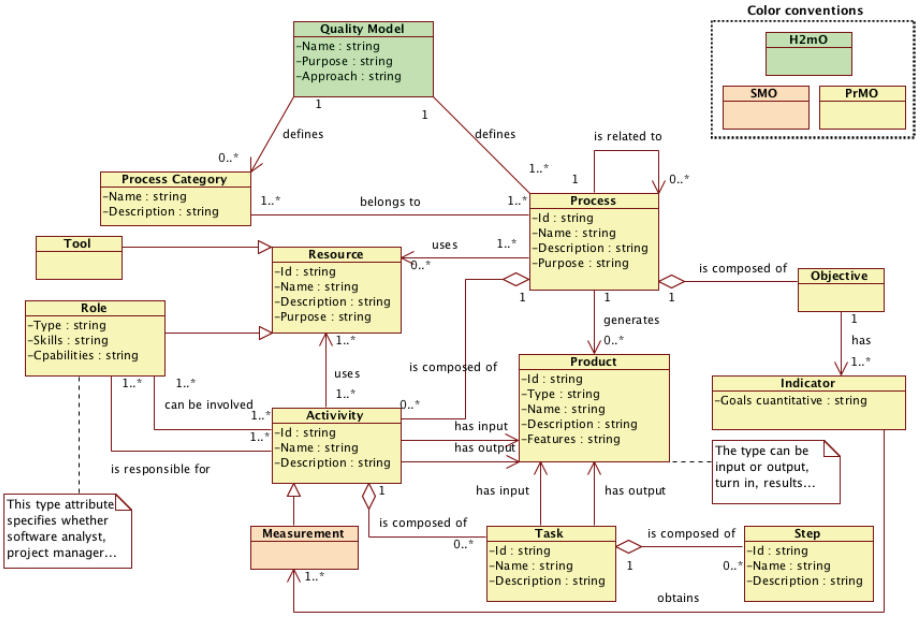


Fig. 1. Representation of PrMO.

As is shown in Figure 1, in general, the hierarchies between concepts represent the fact that in every model all processes in different categories or process groups are grouped together. In the same way, each process is formed by a set of elements or characteristics, such as: activities, tasks, roles, products or artifacts, measurements,

and so on. We do not aim to take in characteristics of all models and existing standards, but rather those that are the most common, as well as those defined in the models analyzed, making its future adaptation and extension possible.

3.2 Instances of PrMO

Currently, the ontology has been applied and used with success in two real cases of application in the context of: (i) a research project in the definition of a unified model for banking sector and a consultancy organization to support the certification of ISO 20000 part 2 (ISO 20000-2) from efforts and institutionalized practices in ISO 27001 certificated companies. Based on PrMO, it was possible to homogenize and build some instances from it and to support the information contained in BASEL II, VAL IT, COBIT, RISK IT, ISO 27002 and ITIL for the first case and ISO 27001 and ISO 20000-2 for the second case. Due to the space limit, this section will focus on showing how the ontology has been instanced and used in two models: CMMI-ACQ and ISO 9001. Another factors such as: the harmonization strategy, homogenization, comparison and integration methods, benefits, findings and harmonization process followed to harmonize the models and standards involved in the case studies are presented in [27] and [28].

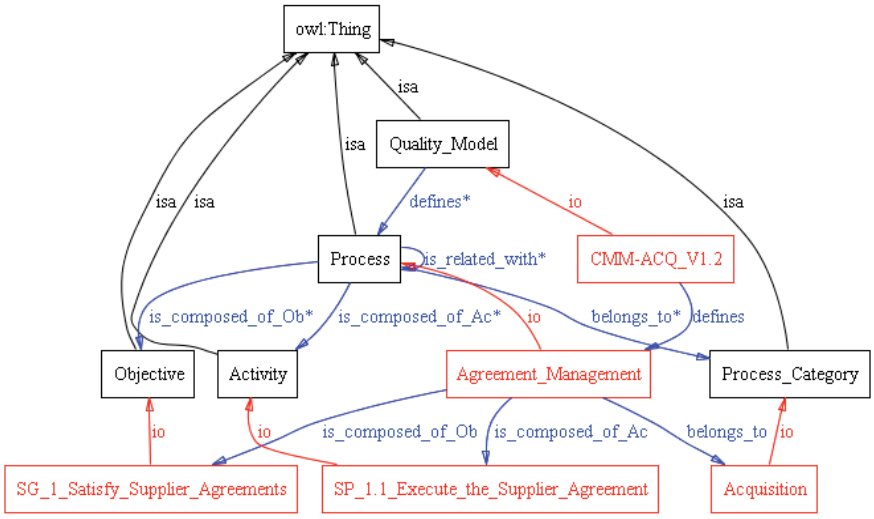


Fig. 2. Instance of CMMI-ACQ V1.2 using PrMO.

Figure 2 and 3 show excerpts of the instances of CMMI-ACQ V1.2 and ISO 9001:2008 using Prótegé-OWL. In Figure 2 it is possible to see that the *Agreement Management* (AM) is a process, which belongs to the *Acquisition Category* of CMMI-ACQ, and AM is composed of an *Objective* (*Specific Goal* (SG) 1 concerning *Satisfy Supplier Agreements*). It is also possible to see the *Specific Practices* (SP) related to this SG. Aiming to improve the understanding of the figure, we have eliminated some concepts, such as *task* and *products* and their nonterminal concept

attributes. Figure 3 shows that *Clause 4 System Quality Management* has been considered as a process of ISO 9001:2000, which belongs to Process Category of the same name. We may also observe one of its activities, *Clause 4.2.3,* concerning control of documents relating to the list in the letters a, b, c, d, e, f, g, the control set needed to carry out this procedure (e.g. approve, review, update documents, and so on). *Clause 4.2.1,* concerning products which the documentation system of quality management should include, are mentioned in the letters a, b, c, d and e of this clause.

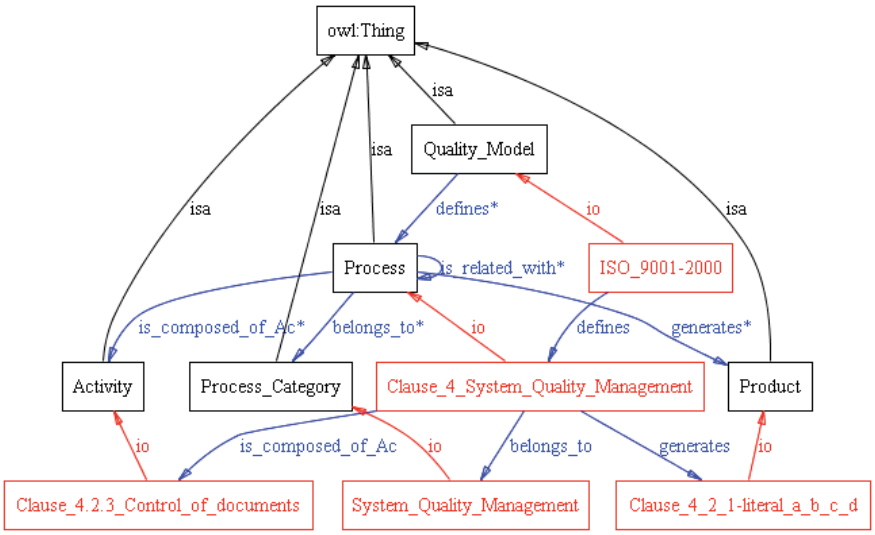


Fig. 3. Instance of CMMI-ACQ V1.2 using PrMO.

PrMO does not only support reference models with a clearly defined process structure but also the models whose elements are organized in less detailed structures. Therefore, PrMO can support other domains such as business where models' architectures may vary (not having such en emphasis on a process dimension) like software, security, amongst others. An example of homogenization of structures of several models by means of PrMO is presented in Table 4. Since each instance (of PrMO) was constructed in the same way, it was possible to map the models by means of similar process elements. CSPE thus allows us to resolve the differences between them and to prepare any reference model before carrying out any comparison, integration or activity concerning harmonization of multiple models.

4 PrMO as Basis to Homogenize Multiple Models

4.1 Common Structure of Process Elements (CSPE)

From process elements defined in PrMO, we have designed a Common Structure of Process Elements or CSPE template, which allows us to have a means of facilitating

and supporting the harmonization of multiple models, through the homogenization of their process structures. CSPE is divided into four sections, which are described next:

- **Section 1: Description (SD1)**. Includes the: process category, process, objectives, activities and related tasks;
- **Section 2: Roles and Resources (SRR2)**. Includes the: resources, tools, roles and Work disciplines defined to carry out the process development, activities or tasks;
- **Section 3: Control (SC3)**. Relates the artifacts, deliverables, results, goals and measurements that serve as verification milestones in the execution of an activity or task;
- **Section 4: Additional Information (SAI4)**. Involves related processes and methods needed to obtain a purpose.

In next sections the HoMethod and its application are presented.

4.2 HoMehtod: A method for homogenization of models

To describe the process elements making use of the proposed structure, we suggest following a homogenization Method (HoMethod). The purpose is to guide the homogenization of multiple models step-by-step. In order to organize and manage the people, activities and steps defined in this method, we define two roles which support their execution: the performers and the reviewers. The activities and tasks, of which HoMethod is made up and which make use of the proposed structure, are presented below:

i). **Acquisition of knowledge about the models involved**. Before carrying out the execution of the harmonization of models, it is suggested that an analysis of each model be carried out, according to some of their elements and/or attributes, e.g. approach, size (number of pages), the development organization,

ii). **Structure analysis and terminology**. The analysis of the structure of a model can turn out to be one of the initial implicit steps in carrying out the implementation or improvement project. Homogenization supports an exhaustive analysis of terminology, syntax and identification of specific words for the models.

iii). **Identification of requirements**. Once the analysis has been done, it is possible to carry out the identification of requirements of software process to be homogenized. That allows us to identify which information of the model will be matched and organized in the structure elements. An example of syntax defined to identify the requirements in the ISO models family is presented in Table 2.

iv). **Carrying out the correspondence**: Such correspondence shows the models reorganized in the four sections of process elements described by CSPE structure. The object of homogenization is to prepare the models for harmonization in multi-model environments.

v). **Analyzing the results**: This activity involves the tasks: resolving the discrepancies of the outcomes of the performers (by reviewers) and verifying and validating these results (by reviewers).

vi). **Presenting the homogenized model**.

Table 2. Syntax to identify the requirements in ISO 20000-2.

Syntax	Descriptions
Shall [verb]	This statement indicates the actions, activities, tasks or procedures that the organization that will develop it will have. It is probable that this statement will be used to describe one or several actions or to derive processes.
Shall [verb] … and [verb]	
Begins with [shall] or shall [verb] that	Identifies a list of derived requirements of processes, procedures, activities or tasks.
Shall be [verb]	Indicates the characteristics associated with a process, or possible roles or work products.
Shall [include]	Indicates the details that the organization must include in a process or work product
Shall be [verb] + [by], [to] or [on]	This syntax helps to identify the detail of some procedures or processes.
Documented, input, output	Indicates a possible work product. It might includes some characteristics related to the work product.

5 Application of CSPE

In this section, we describe the steps carried out for the homogenization of models and requirements contained in ISO 20000-2. Table 3 shows an example of homogenization of clause 6.5 of ISO/IEC 20000-2 using the CSPE template and its application employing HoMethod.

5.1 Homogenization of ISO 20000-2

We will now give a brief summary of the application of the steps described, implementing the common structure in homogenization of the ISO 9001:2000 standard. The semantic analysis of the standard was carried out in the same way as other authors such as Paulk [6] and Mutafelija & Stromber [7] have done, where the requirements are identified by analysing the "Shall" and "Should" statements. Based on a syntax table to identify the requirements in ISO 9001 defined in [31], analysis and identification of both requirements ISO 9001 and ISO 20000-2 were carried out. This syntax analysis allowed us to identify the practices required by the standards better, thereby decreasing a large part of the ambiguity and subjectivity that is an integral part of trying to understand them. Table 2 shows the syntax used to identify the requirements in ISO 20000-2; it has been extended and updated from syntax defined in [31] which did not include the analysis of input or output statements and clauses as possible work products. These are described in all ISO standards.

An example of the result of the homogenization is shown in Table 3. On the table, clause 6.5, related to capacity management defined in ISO 20000-2, is organized and structured according to the CSPE Template. In this table we can see that not all the elements of the four sections of the common structure found any correspondence. This is because the standard "doesn't define" or set out detailed information for that correspondence.

Table 3. Homogenization of clause 6.5 defined in ISO 20000-2.

Process 6.5 Capacity management				
SD1.1. Process Category		6. Service Delivery Processes		
SD1.2 Processes	*ID:*	6.5	*Name:*	Capacity management
	Goal	To ensure that the organization has, at all times, sufficient capacity to meet the current and future agreed demands of the business.		
SD1.4. Activity	**SD1.5. Task**			**SC3.1. Artifacts**
The Clause 6.5 refers to the capacity management	1. The current and expected requirements of the business in relation to service should be known, in terms of what the business is going to need for it to be able to give a service to its clients. 2. The business forecasts and estimates of workload should be translated to specific requirements and must be documented. 3. The result of changes in workload or environment should be predictable. 4. Current and historical data of the use of components and resources should be collected and analyzed at the appropriate level to support the process. 5. Management capacity should be the focal point for all issues of performance and capacity. 6. The process should provide direct support to the development of new services and modifications to these, performing a sizing and modelling service. 7. A capacity plan must be generated and this should be prepared annually, at least. 8. A good understanding should exist of the technical infrastructure and its present and projected capabilities.			1. Capacity plan that documents the actual performance of the infrastructure and the expected requirements, often enough to take into account the rate of change of services and service volumes, information reports and change management in the client's business. 2. Documentation with the existing options, along with the cost involved in meeting the business requirements and solutions recommended for achieving the service level objectives.
SAI4.1 Related processes	Clause 6.5 is related to clauses 6.1, 7.2 y 9.2.			

ISO 20000-2 neither defines nor documents clearly many of the requirements that it suggests should be put into operation (for example activities, tasks and artifacts). Correspondence and formalization of the information presented in it with regard to process elements of structure had made it more possible to understand the requirements associated with it. An example is the identification and correspondence of activities, tasks and artifacts. For greater detail about the original descriptions of models analyzed, we suggest the corresponding reference be consulted.

The proposed structure has also been applied to other models and standards, such as CMMI (DEVelopment and ACQuisition), ISO 9001, Cobit 4.1, ITIL, Risk IT, Val IT, BASEL II, ISO 27001, ISO 27002, ISO 20000-2, PMBOK, and MoProSoft, see [27, 31, 32].

5.2 Homogenization through a supporting tool

Within the ontology groundwork, we designed and developed one of the functionalities of HProcessTOOL [33], which is a web tool to manage harmonization projects by supporting specific techniques. It also supports the management which controls and monitors the resulting harmonization projects. When a user logs on to HProcessTOOL, s/he can harmonize the models involved in a harmonization project through CSPE, which, as discussed earlier, is a template based on PrMO which takes some process elements defined in it and provides a way to support the harmonization of reference models.

Our tool has been used successfully in case studies presented earlier, see [33]. We have thus been able to validate and demonstrate that PrMO can be used on a WEB platform. We can also say that, given the generality of PrMO, it has not been necessary to use the mechanism of inheritance and restriction to homogenize multiple models. However, since each model uses different names to appoint its process elements or simply because some of them aren't defined, we have had to establish a correspondence table with regard to the process elements defined in the ontology. Currently, we have homogenized some models and standards through OPrM such as CMMI (DEVelopment and ACQuisition), ISO 9001,COBIT 4.1, ITIL, RISK IT, VAL IT, BASEL II, ISO 27001, ISO 27002, ISO 20000-2, PMBOK, and MoProSoft. Table 4 shows the table of correspondence used and an example to homogenize the process elements of some reference models such as CMMI (DEV and ACQ), ISO (9001, 27001, 20000-2) and COBIT.

Table 4. Correspondence of models according to OPrM.

PE of CSPE	**CMMI-DEV CMMI-ACQ CMMI-SVC** **Example: CMMI-ACQ V1.2**	**ISO 9001:2008 ISO 27001:2005 ISO 20000-2:20011** **Example: ISO 9001:2008**	**COBIT 4.1**
Process Category	*Categories,* e.g. Support, Engineering, Process and Project Management.	*Requirements*, e.g. System of Quality Management.	*Domains*, e.g. Plan and organize.
Process	*Process Areas*, e.g. Agreement Management from CMM-ACQ.	*Principal Clauses*, e.g. clause 4 concerning System of Quality Management.	*Process*, e.g. PO1 concerning define a strategic IT plan.
Objective	*Specific Goal (SG),* e.g. SG 1 Satisfy Supplier Agreements	*Inherent Information*	*Inherent Information*
Activity	*SpecificPractices,* e.g. Specific Practice 1.1 Execute the Supplier Agreement.	*Subclauses (II[b]),* e.g. clause 4.1 concerning the general requirements.	*Activities,* e.g. PO1.1 IT value Management.
Task	*SCiSP[a],* e.g. numeral 5 concerning Monitor risks involving the supplier.	*Information Not found*	*Information Not found*
Artifact or Product	*Information Not found*	*Clause 7.3.4,* e.g. Participants in such reviews shall include representatives of functions concerned with the design and development stage(s) being reviewed.	*Rol &Responsibility Chart (RACI),* e.g. Business Executive role as responsible for: linking business goals to IT goals.
Role	*Information Not found*	*Clause 6.3.* e.g. nfrastructure includes, as applicable, a) buildings, b) process equipment (both hardware and software), and, c) supporting services	*Information Not found*
Resource	*Information Not found*	*Information Not found*	*Information Not found*
Tool	*Typical Work Products and,* e.g. Integrated list of issues.	*Subclauses (II[b]),* e.g. Clause 4.2.1, describes the term "documented procedure" as referring to a procedure that must be supported by processes to establish it, document it, implement it and maintain it.	*Outputs,* e.g. Strategic IT plan which must be obtained in aPO1.1 activity.
Measure	*Information Not found*	*Information Not found*	*Metrics,* e.g. to measure degree of approval of the IT strategic/tactical plans on the part of business owners.

a. SCiSP: Subpractices Contained in Specific Practices, b. II: Inherent Information

Other applications of PrMO are as follows:

- The CSPE is being used to develop a functionality which means the user can (design, construct, apply and analyze) make appraisals from reference models stored in HProcessTOOL. As it will be supporting reference models stored through HProcessTOOL and CSPE, it will be flexible enough to support process appraisals in the context of global software development and adaptable to possible changes that may occur with such models. In that sense, it could be a useful tool, making quality assessment and improvement of the organizations' processes possible at a global level.
- CSPE has demonstrated that it could be useful as a way of supporting the assessment of structural differences and of finding out the level of detail of the reference models involved in a harmonization project. This allows us to identify an initial set of differences that has to be solved before carrying out a mapping process.

6 Conclusions and Future Work

In this paper, PrMO has been presented, this being an ontology of process-reference models designed to facilitate the harmonization of multiple models and standards. It has been illustrated how PrMO has been instanced in a clause of ISO 20000-2. Using the ontology, we have developed a functionality, which, through a Common Structure of Process Elements CSPE, makes it possible to support the homogenization of structural differences found between reference models. It is part of a web tool called HProcessTOOL. We should also add that we are currently developing an appraisal tool, which permits the design, construct, application and analysis of assessments of an organization to be performed from the homogenized models and stored in HProcessTOOL.

The homogenization of models is currently a manual task, so, as future work, the next step in this study will involve the automation of the homogenization stage through development of specific algorithms which let us extend the capability of our tools. It is not our intention to automatize all the tasks and activities involved. We do, however, wish to help users with an automatic step during mapping or correspondence of process elements to our CSPE.

It should also be said that, since PrMO has been used to instance different process and reference models, it has shown that it can also be used as a basis for supporting the design and building of an organization's processes. That being the case, we hope to develop a functionality to support the definition of organizations' processes through our ontology and tool. The information stored will be able to be used as a benchmark of processes for other organizations, as well as to help them during definition of their own processes.

Although PrMO has been applied in the homogenization of several models, in the quest to cover a wider range of needs, we hope to extend models and standards modeled through PrMO and stored in HProcessTOOL.

Acknowledgments. This work has been funded by the projects: ARMONÍAS (JCCM of Spain, PII2I09-0223-7948), PEGASO/MAGO (MICINN and FEDER of Spain, TIN2009-13718-C02-01) and QUASIMODO (PAC08-0157-0668). Acknowledgements by Francisco J. Pino to the University of Cauca where he works as Associate Professor.

References

1. Pardo, C., Pino, F.J., García, F., Piattini, M., Baldassarre, M.T.: Trends in Harmonization of Multiple Reference Models: In: Evaluation of Novel Approaches to Software Engineering, CCIS. Springer-Verlag, (Special edition best papers ENASE 2010, extended and updated paper) (2011) 61–73
2. Wang, Y., King, G.: Software Engineering Processes: Principles and Applications. CRC Press (2000)
3. SEI: The PrIME Project. (2010). Available on http://goo.gl/p2GX3
4. SPICE: Enterprise SPICE. An enterprise integrated standards-base model (2008). Available on http://www.enterprisespice.com/
5. ITGI: Aligning Cobit 4.1, ITIL V3 and ISO/IEC 27002 for Business Benefit. Technical report, IT Governance Institute (ITGI) and Office of Government Commerce (OGC). (2008)
6. Paulk, M.C.: How ISO 9001 compares with the CMM. IEEE Software. 12, 74–83 (1995)
7. Mutafelija, B., Stromber, H.: Systematic Process Improvement Using ISO 9001:2000 and CMMI (2003)
8. Gokhan Halit, S., Mieczyslaw, M.K.: An OWL Ontology for Representing the CMMI-SW Model. Technical report, (2010)
9. Liao, L., Qu, Y., Leung, H.K.N.: A Software Process Ontology and Its Application. In: 4th International Semantic Web Conference, pp. Galway (2008)
10. Salviano, C.F., Figueiredo, A.M.C.M.: Unified Basic Concepts for Process Capability Models. In: SEKE, pp. 173-178. (2008)
11. Ferchichi, A., Bigand, M., Lefebvre, H.: An Ontology for Quality Standards Integration in Software Collaborative Projects. In: First International Workshop on Model Driven Interoperability for Sustainable Information Systems, pp. 17–30. Montpellier (2008)
12. Malzahn, D.: Assessing - Learning - Improving, an Integrated Approach for Self Assessment and Process Improvement Systems. Proceedings of the 2009 Fourth International Conference on Systems. IEEE Computer Society (2009) 126-130
13. Mendes, O., Abran, A.: Software engineering ontology: A development methodology. In: Metrics News, pp. 68-76. Québec (2004)
14. Pardo, C., Pino, F.J., García, F., Piattini, M., Baldassarre, M.T.: An ontology for the harmonization of multiple standards and models. Computer Standards & Interfaces. 34, 48-59 (2012)
15. OMG: Software & Systems Process Engineering Meta-Model Specification. SPEM 2.0. Technical report, (2008)
16. Cugola, G., Ghezzi, C.: Software Processes: a Retrospective and a Path to the Future. Software Process: Improvement and Practice. 4, 101-123 (1998)
17. Derniame, J.-C., Kaba, A.B., Warboys, B.: The Software Process: Modelling and Technology: In: Montenegro, C. (ed.): Software process: principles, methodology, and Technology. Springer, Germany (1999) 1-12
18. Fuggetta, A.: Software process: A Roadmap. In: International Conference on Software Engineering (ICSE), pp. 25-34. Limerick (2000)

19. Benali, K., Derniame, J.C.: Software process modeling: What, who and when. Software Process Technology, Lecture Notes in Computer Science. (1992)
20. Finkelstein, A., Kramer, J., Nuseibeh, B.: Software process modelling and technology. Advenced Software Development Series. 3, (1994)
21. McChesney, I.: Toward a classification scheme for software process modelling approaches. Information and Software Technology. 37, 363-374 (1995)
22. Fuggetta, A., Wolf, A.L.: Software process. Trends in Software. 4, 89-100 (1996)
23. Huff, K.: Software process modeling. Trends in Software: Software Process. (1996)
24. García, F., Bertoa, M., Calero, C., Vallecillo, A., Ruiz, F., Piattini, M., Genero, M.: Towards a consistent terminology for software measurement. Information & Software Technology. 48, 631-644 (2006)
25. Tautz, C., Gresse Von Wangenheim, C.: REFSENO: A representation formalism for software engineering ontologies. Technical report, Fraunhofer IESE-Report No. 015.98/E V1.1 (1998)
26. Protégé: The Protégé Ontology Editor and Knowledge Acquisition System. (2010). Available on http://protege.stanford.edu
27. Pardo, C., Pino, F.J., García, F., Piattini, M., Baldassarre, M.T., Lemus, S.: Homogenization, Comparison and Integration: A Harmonizing Strategy for the Unification of Multiple-Models in the Banking Sector. In: The 12th International Conference on Product Focused Software Development and Process Improvement (PROFES 2011), pp. 59-72. Visaggio, G., Caivano, D., Oivo, M., Baldassarre, M.T., Bari (2011)
28. Pardo, C., Pino, F.J., García, F., Piattini, M., Baldassarre, M.T.: A Process for Driving the Harmonization of Models. In: The 11th International Conference on Product Focused Software Development and Process Improvement (PROFES 2010). Second Proceeding: Short Papers, Doctoral Symposium and Workshops, pp. 51-54. Markku Oivo, M.A.B., Matias Vierimaa, Limerick (2010)
29. Derniame, J.-C., Kaba, B., Wastell, D.: Software Process: Principles, Methodology and Technology. Lecture Notes in Computer Science 1500. 307 (1999)
30. Acuña, S.T., Antonio, A.D., Ferré, X., López, M., Maté, L.: The Software Process: Modelling, Evaluation and Improvement: In: Chang, I.S.K. (ed.): Handbook of Software Engineering and Knowledge Engineering, Vol. 1 Fundamentals. World Scientific, New Yersey (EE.UU) (2001) 193-237
31. Pardo, C., Pino, F., García, F., Piattini, M.: Homogenization of Models to Support multi-model processes in Improvement Environments. In: 4th International Conference on Software and Data Technologies ICSOFT'09, pp. 151-156. Sofía (2009)
32. Pardo, C., Pino, F.J., García, F., Piattini, M., Rosado, J.: Armonizando ISO/IEC 20000 e ISO/IEC 27001 para integrar la gestión de servicios y la seguridad de la información. In: XV Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2010), pp. 225-235. Teniente, E., Abrahão, S., Valencia (2010)
33. Pardo, C., Pino, F.J., García, F., Romero, F.R., Piattini, M., Baldassarre, M.T.: HProcessTOOL: A Support Tool in the Harmonization of Multiple Reference Models: V. In: B. Murgante, O.G., A. Iglesias, D. Taniar, B. Apduhan (ed.): ICCSA 2011 Proceedings, LNCS, Vol. 6786. Springer, Santander (2011) 370-382

An Approach to Test-Driven Development of Conceptual Schemas*

Albert Tort, Antoni Olivé and Maria-Ribera Sancho

Department of Service and Information System Engineering
Universitat Politècnica de Catalunya - BarcelonaTech
Barcelona, Spain

{atort,olive,ribera}@essi.upc.edu

1 Summary

Conceptual modeling is an essential requirements engineering activity. Its objective is the development of the conceptual schema (CS) of an Information System (IS), which defines the general knowledge that an IS needs to know to perform its functions.

A CS consists of a structural (sub)schema and a behavioral (sub)schema. The structural schema consists of a taxonomy of entity types, a set of relationship types, and the constraints they must satisfy. The behavioral schema consists of a set of event types with their characteristics, constraints and effects.

A CS has semantic quality when it is valid and complete. Validity means that the schema is correct (the knowledge it defines is true for the domain) and relevant (the knowledge it defines is necessary for the system). Completeness means that the conceptual schema includes all relevant knowledge. Ensuring that a CS has semantic quality is a fundamental goal for its validation. This goal can be achieved by checking that the knowledge the system requires to know according to stakeholders' expectations is the same as the knowledge defined by the conceptual schema.

We report the development of a novel conceptual modeling method, named Test-Driven Conceptual Modeling (TDCM), which fosters the achievement of the above mentioned quality properties. TDCM is a test-driven method for the incremental development of conceptual schemas by continuous validation. As far as we know, this is the first work that explores the use of testing to drive conceptual modeling.

TDCM is based on our approach to test executable conceptual schemas written in formal modeling languages like UML/OCL. In this approach, automated conceptual test cases specify executable scenarios and include assertions that formalize expectations about them. In TDCM, the *Conceptual Schema Under Development* (CSUD) is developed in short iterations by applying the following tasks: (1) Write a test case that formalizes a story which is expected to be feasible with the knowledge defined in the schema; (2) change the schema to pass the test case; and (3) refactor the schema to improve the specification without changing the defined knowledge.

* This work has been partly supported by the Ministerio de Ciencia y Tecnología and FEDER under the project TIN2008-00444, Grupo Consolidado.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 407-408, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

An iteration starts by adding a *new test case* to the passing test set of the previous iteration (*previous test set*). The objective of each iteration is to change the schema so that it includes the knowledge to correctly execute the *new test case*. The *previous test set* including the new test case is the *current test set* of the iteration. An iteration can only finish when the overall verdict of the *current test set* is *Pass*. At this point, the CSUD includes the knowledge to execute the tested cases as expected.

TDCM fosters refactoring of the CSUD in order to improve the quality of the conceptual schema specification without changing the knowledge specified in it. If the verdict of the *current test set* ceases to pass after refactoring, then we realize that the knowledge of the schema has not been preserved. If the verdict remains *Pass* and no more refactoring is felt to be needed, new iteration can be initiated.

Table 1 summarizes the interpretation of the verdict of the *current test set* (CTC) and the regression test set (RTS) at each iteration. This interpretation drives the application of TDCM in order to evolve the CSUD.

Verdict	Current test case (CTC)	Regression test set (RTS)	
Error	Relevant knowledge needs to be added to the CSUD	*Schema element removed*	The CSUD becomes incomplete and the deleted schema element needs to be restored
		Relevant constraint added	Modify inconsistent states to maintain consistent test cases
Fail	The knowledge defined in the CSUD needs to be corrected according to the asserted expectations	*Both CTC and RTS may Pass without changing them (only changing the CSUD)*	The knowledge defined in the CSUD needs to be corrected
		Neither CTC nor RTS can Pass without changing them	Inconsistent requirements
Pass	The CSUD has the necessary knowledge and satisfies the asserted expectations formalized in CTC	The CSUD still has the necessary knowledge and satisfies the asserted expectations formalized in RTS	

Table 1. Pass/Error/Failure interpretation

The proposed method is applicable to different kinds of projects and may be integrated into existing software development methods when they are based on iterative paradigms and they include artifacts to specify conceptual schemas. TDCM can also be used even if the conceptual schema to be developed is the main purpose of the project. Using TDCM, conceptual modelers have at any time fully tested schemas.

References

1. Tort, A., Olivé, A., Sancho, M. R.: An Approach to Test-Driven Development of Conceptual Schemas. Data & Knowledge Engineering, 69(6), 598-618 (2011)

Requirement-based Approach for Groupware Environments Design[1]

Víctor M. R. Penichet, María D. Lozano, José A. Gallud, Ricardo Tesoriero

Universidad de Castilla-La Mancha. Departamento de Sistemas Informáticos
Av. España S/N, Campus Universitario de Albacete, (02071), Albacete, España
{victor.penichet, maria.lozano, jose.gallud,
ricardo.tesoriero}@uclm.es

1 Summary

Groupware applications have special features that, if they were taken into account from the very beginning, could reasonably improve the quality of the system. Such features concern human-computer-human interactions, i.e., a further step in the human-computer interaction field: communication, collaboration, cooperation and coordination, time, space, and awareness are issues to be considered. This paper presents a novel approach to gather requirements for groupware applications. The proposal is based on a methodology that includes the use of templates to gather the information regarding the different types of requirements. The requirements templates have been extended to include new information to give account of specific features concerning groupware applications. The information gathered is managed in a CASE tool we have developed; then general and specific diagrams are automatic or semi-automatically generated.

2 Templates and Metadata to Describe Requirements

We have based our proposal on the Amador Duran's process model (Duran, 2000) extending some of the templates proposed in this work, with some other metadata we consider important for the specification of groupware applications. Besides, new templates have been defined following the same purpose.

We propose a template to gather the information regarding objectives and requirements which consists of three parts: a general template with the common metadata concerning both objectives and requirements. Then a specific extension with different metadata to specify objectives and requirements of the system (different specific extensions to specify objectives, functional requirements and non-functional requirements), and finally a CSCW extension with metadata regarding groupware

[1] We would like to acknowledge the project with reference CICYT TIN2011-27767-C02-01 and the Junta de Comunidades de Castilla-La Mancha PEII09-0054-9581 and PII2C09-0185-1030 projects for partially funding this work.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 409-410, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

issues in case it is necessary. Every system objective is specified by way of a Use Case Diagram. The most important use cases will be described by means of such requirements templates.

Templates are also used to describe participants (*actors*, *groups*, *individuals*, *users* and *agents*) and the tasks to be performed by the participants as well as the roles they will play.

3 Automatic and Semi-automatic Models Generation from Requirements Information

In this paper we have presented a proposal to design groupware environments starting from the requirements specification and taking into account the particular features of this kind of systems. This way, these special features have been accurately analyzed and then taken into account from the very beginning of the development process, just in the first stage of the software life cycle: the requirements gathering stage.

These special features have been considered and included in different templates proposed to describe the requirements of the system. The templates and the metadata we propose, provide information about the system objectives, the information requirements, the non-functional requirements, the functional requirements, the organizational structure of the participants in a system, and the participants of the system.

Some examples of use of these templates and diagrams have been depicted to show the utility of them when developing groupware applications. Other aspect to highlight is the development of a CASE tool, the TOUCHE CASE Tool, to support the methodology proposed. This tool allows the automatic generation of some models taking into account the aforementioned information. Data regarding the participants of the system and the functional requirements' normal sequence and their relationship with objectives help on the generation of every use case diagram. The Organizational Structure Diagram (OSD), which describes how the different organizational items are related to constitute the structure of groups, roles, and actors, may be also automatically generated from the information regarding participants and roles. Some other diagrams such as the task diagram and the co-interaction diagram retrieve information from these templates to depict a first approach of the system.

References

1. Víctor M. R. Penichet, María D. Lozano, José A. Gallud, Ricardo Tesoriero: Requirement-based Approach for Groupware Environment Design. Journal of Systems and Software (JSS). Volume 83. Issue 8, Elsevier Science Inc. New York, NY, USA. ISSN: 0164-1212. DOI: 10.1016/j.jss.2010.03.029. August, 2010

GAmeraHOM: una herramienta de generación de mutantes de orden superior para WS-BPEL

Emma Blanco Muñoz, Antonio García Domínguez, Juan José Domínguez Jiménez e Inmaculada Medina Bulo

Universidad de Cádiz, Escuela Superior de Ingeniería
C/Chile 1, CP 11002, Cádiz, España,
emma.blancomu@alum.uca.es
{antonio.garciadominguez,juanjose.dominguez,inmaculada.medina}@uca.es

Resumen El desarrollo de técnicas para que la prueba de mutaciones disminuya el tiempo de cómputo y reduzca los mutantes producidos no han ido en paralelo al desarrollo de herramientas generadoras de mutantes que las implementen. Una de las técnicas de optimización propuestas sugiere utilizar mutantes de orden superior, sin embargo apenas existen herramientas que la implementen. En este trabajo se presenta GAmeraHOM, el primer generador de mutantes de orden superior para WS-BPEL, basado en la mutación evolutiva mediante el empleo de un algoritmo genético. Esto conlleva una reducción del número de mutantes generados y ejecutados, seleccionando principalmente los mutantes difíciles de matar y los potencialmente equivalentes. Este conjunto reducido permitirá mejorar la calidad del conjunto de casos de prueba. La herramienta GAmeraHOM provee una configuración fácilmente parametrizable y además permite adaptar el código del generador a otros lenguajes de forma independiente.

Palabras clave: prueba de mutaciones, mutantes de orden superior, generador de mutantes, algoritmo genético, WS-BPEL 2.0, composiciones de servicios, prueba del software

1. Introducción

La prueba de mutaciones es una técnica de prueba del software de caja blanca basada en errores que ha sido usada en muchos lenguajes de programación para evaluar conjuntos de casos de prueba. Sin embargo, presenta el inconveniente de tener un alto coste computacional. Debido a esto, han surgido diversas técnicas para reducir el número de mutantes generados y el tiempo de ejecución.

En cuanto a las técnicas existentes para reducir el número de mutantes generados, destacamos la prueba de Mutación Evolutiva [2] basada en un algoritmo evolutivo que no genera todos los mutantes, sino solamente un subconjunto reducido conforme vaya necesitando el proceso de selección.

Esta técnica fue implementada en GAmera [3], una herramienta para la generación y selección de mutantes WS-BPEL mediante algoritmos genéticos.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 411-414, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

Otra de estas técnicas fue propuesta por Yue Jia en [5], donde amplió el concepto de mutación simple para que un programa original pudiera sufrir una o más modificaciones dando lugar a *mutantes de orden superior* (HOM). Aunque se demostró la efectividad de este método, solo la herramienta MILU [4] produce mutantes de orden superior para programas C.

En este trabajo presentamos GAmeraHOM, el primer generador de mutantes de orden superior para WS-BPEL que usa la técnica de mutación evolutiva.

2. GAmeraHOM

Uno de nuestros objetivos ha sido desarrollar un generador de mutantes de orden superior lo más genérico posible, de modo que no sufriera grandes cambios al aplicarlo a un lenguaje de programación distinto. Por ello, este proyecto queda dividido en tres componentes principales:

- Una API con el modelo de datos e interfaces genéricas.
- Un núcleo donde se desarrolla la lógica del algoritmo genético y el lanzador por línea de órdenes.
- Un soporte para el lenguaje que aporta los operadores de mutación y el sistema de ejecución de mutantes.

Con esta distribución aseguramos que el núcleo no dependa del lenguaje en el que queramos generar mutantes y que este únicamente dependa de la API.

Solamente existe un componente dependiente del lenguaje, por lo que sería este el que habría que reemplazar para adaptar esta herramienta a otro lenguaje.

GAmeraHOM está preparado para WS-BPEL y está basada en MuBPEL [6], una herramienta creada previamente por el mismo grupo de investigación para la ejecución y comparación de mutantes WS-BPEL de primer orden.

Hemos utilizado la técnica de la mutación evolutiva, ya usada previamente en GAmera [3,2]. En [1] se publicó la arquitectura de GAmeraHOM, por lo que este trabajo servirá para mostrar la herramienta ya desarrollada.

El generador está escrito en Java, usa Maven como sistema de compilación y se ha hecho especial hincapié en la calidad del software apoyándonos en herramientas como Sonar y Jenkins.

Para parametrizar la configuración usamos un fichero en formato YAML, único argumento del programa. En él podemos indicar el tamaño máximo de la población del algoritmo genético, el máximo orden de mutación, el motor de ejecución (indicando las rutas al conjunto de casos de prueba y al programa WS-BPEL original), los operadores genéticos, los generadores de individuos, los operadores de selección y los criterios de parada, entre otras opciones.

GAmeraHOM funciona bajo línea de órdenes. En el listado 1 pueden observarse las múltiples opciones que disponemos para ejecutarlo.

fitnessmap Genera un informe de un registro de simulación sobre la distribución del fitness de todos los individuos de primer orden.

randrun Se basa en la selección aleatoria e intenta imitar el comportamiento del algoritmo genético con la información del fichero de configuración.

```
$ gamerahom
GAmera version 2.0-SNAPSHOT (2012.06.30 05:44:28)

Available subcommands:

 * fitnessmap simrecord.(raw|yaml)
 * randrun config.yaml
 * randsimrun config.yaml simrecord.yaml
 * run config.yaml
 * simrecord config.yaml simrecord.yaml
 * simrun config.yaml simrecord.yaml

For more help about a subcommand, use 'gamerahom subcmd -h'
```

Listado 1. Opciones de ejecución

```
MUTANT_N {ORDER (OPERATOR, LOCATION, ATTRIBUTE)+},
    [COMPARISON RESULTS]
MUTANT_1 {3 (ECN,1,4), (EIU,1,0), (ASI,0,1)},
    [1 1 1 0 0 1]
MUTANT_2 {2 (ERR,2,3), (AIE,0,0)},
    [1 0 0 1 1 0]
MUTANT_3 {4 (AIE,1,1), (ECN,1,1), (ECN,1,2), (CDE,1,2)},
    [1 1 1 0 0 1]
```

Listado 2. Fragmento del *salón de la fama*

randsimrun A la opción del *randrun* se le añade el reemplazo del sistema de ejecución de mutantes por una simulación en base a un registro de simulación creado previamente con *simrecord*. Esto acelerará las pruebas.

run Esta es la instrucción básica que se encarga de ejecutar el algoritmo genético con la configuración indicada en el fichero YAML.

simrecord Crea un registro de simulación con datos como resultados de la comparación de las pruebas para los mutantes y el programa original, el tiempo total empleado en las pruebas, etc.

simrun Carga el algoritmo genético con el fichero de configuración usando en lugar del sistema de ejecución por defecto, la información proporcionada por el registro de simulación.

En el listado 2 se muestra parte del *salón de la fama* que se produce en una de las pruebas de ejecución del algoritmo genético. Podemos observar todos los mutantes que se han creado en las generaciones del algoritmo genético, indicando cómo se obtienen a partir del programa original aplicando los operadores de mutación correspondientes.

Además podemos ver los resultados de las comparaciones de las salidas de las pruebas de los mutantes y del programa original: cero si las salidas son iguales (el caso de prueba no es capaz de detectar el mutante), uno si las salidas son distintas (ahora el caso de prueba sí puede detectar el mutante).

Analizando los resultados obtenidos podemos detectar aquellos mutantes potencialmente equivalentes y difíciles de matar, es decir aquellos mutantes que han producido la misma salida que el original ante las pruebas, y que son de nuestro interés para así poder generar nuevos casos de pruebas que los detecten.

GAmeraHOM está disponible para Linux y la distribución de la herramienta puede encontrarse ejecutando el guión alojado en `https://neptuno.uca.es/redmine/projects/sources-fm/repository/raw/trunk/scripts/install.sh`.

3. Conclusiones y trabajos futuros

En este trabajo se presenta GAmeraHOM, el primer generador de mutantes de orden superior para composiciones WS-BPEL. Una vez diseñada la herramienta realizaremos experimentos para valorar la calidad de los mutantes generados y por consiguiente, la búsqueda de los valores óptimos de configuración.

Agradecimientos

Este trabajo ha sido financiado por el proyecto MoDSOA (TIN2011-27242) del Programa Nacional de Investigación, Desarrollo e Innovación del Ministerio de Ciencia e Innovación y por el proyecto PR2011-004 del Plan de Promoción de la Investigación de la Universidad de Cádiz.

Referencias

1. Blanco-Muñoz, E., García-Domínguez, A., Domínguez-Jiménez, J.J., Medina-Bulo, I.: Propuesta de una arquitectura para la generación de mutantes de orden superior en WS-BPEL. In: Actas de las XVI JISBD (2011)
2. Domínguez-Jiménez, J., Estero-Botaro, A., García-Domínguez, A., Medina-Bulo, I.: Evolutionary Mutation Testing. Information and Software Technology 53(10), 1108–1123 (2011)
3. Domínguez-Jiménez, J.J., Estero-Botaro, A., García-Domínguez, A., Medina-Bulo, I.: GAmera: A Tool for WS-BPEL Composition Testing Using Mutation Analysis. In: ICWE. pp. 490–493. Viena, Austria (2010)
4. Jia, Y., Harman, M.: MILU: A Customizable, Runtime-Optimized Higher Order Mutation Testing Tool for the Full C Language. In: TAIC-PART '08. pp. 94–98. IEEE (2008)
5. Jia, Y., Harman, M.: Higher Order Mutation Testing. Information and Software Technology 51(10), 1379–1393 (2009)
6. UCASE Research Group: MuBPEL. `https://neptuno.uca.es/redmine/projects/sources-fm/wiki/MuBPEL` (2012)

MuBPEL: una Herramienta de Mutación Firme para WS-BPEL 2.0

Antonio García Domínguez, Antonia Estero Botaro, Juan José Domínguez Jiménez, Inmaculada Medina Bulo y Francisco Palomo Lozano

Universidad de Cádiz, Escuela Superior de Ingeniería,
C/Chile 1, CP 11002, Cádiz, España,
{antonio.garciadominguez, antonia.estero, juanjose.dominguez,
inmaculada.medina}@uca.es

Resumen La prueba de mutaciones evalúa la calidad de un conjunto de casos de prueba realizando cambios en el programa y comprobando si las pruebas los detectan. Se ha aplicado con éxito sobre programas FORTRAN, C y Java, entre otros. Dos problemas comunes son la dificultad de detectar cambios que no modifican el significado del programa, y su alto coste computacional. La mutación firme es una variante de la mutación fuerte tradicional que compara estados intermedios. En este trabajo presentamos MuBPEL, una herramienta de código abierto de mutación firme para composiciones de servicios Web escritas en WS-BPEL 2.0. Los operadores de mutación están implementados en XSLT 2.0 y su organización permite añadir nuevos operadores de forma sencilla. MuBPEL integra el motor ActiveBPEL y la biblioteca de pruebas unitarias BPELUnit de forma transparente y permite paralelizar el trabajo.

Palabras clave: prueba de mutaciones, mutación firme, WS-BPEL 2.0, composiciones de servicios, prueba del software

1. Introducción

La técnica de prueba de mutaciones permite evaluar la calidad de un conjunto de casos de prueba. Consiste en introducir ligeros cambios o *mutaciones* en un programa a través de un conjunto de *operadores de mutación* específicos del lenguaje de programación, generando *mutantes*. Un buen conjunto de casos de prueba debería distinguir al programa original de los mutantes.

La prueba de mutaciones se ha aplicado a numerosos lenguajes de programación, como C [2] o FORTRAN [6]. Su principal ventaja es que considera el comportamiento del programa y no sólo su estructura, como ocurre con las métricas habituales de cobertura del código. Un mutante no detectado (no *matado*) por las pruebas puede indicar de forma concreta qué caso de prueba nos falta. Sin embargo, la prueba de mutaciones tiene sus desventajas: no es posible saber de forma automática si un mutante no detectado es equivalente al programa original, y ejecutar todos los mutantes puede llevar mucho tiempo. Por estas razones, se han propuesto diversas variantes sobre la mutación fuerte tradicional,

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 415-418, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

que compara las salidas finales de los programas. Por ejemplo, la mutación débil compara el estado interno justo tras la mutación, y la mutación firme compara varios estados intermedios. Estas variantes se pueden combinar con otras técnicas de reducción de costes, como el uso de múltiples máquinas o procesadores, o la selección manual de algunos de los operadores (*mutación selectiva*).

Por otro lado, el Web Services Business Process Execution Language 2.0 (WS-BPEL 2.0) [8] permite implementar composiciones de servicios Web a muy alto nivel. Es un lenguaje imperativo basado en XML con construcciones específicas para el manejo de eventos, la gestión de concurrencia y la compensación ante fallos, etc. Debido a su creciente popularidad, Estero Botaro et al. han propuesto un conjunto de operadores de mutación para WS-BPEL 2.0 [4].

En este trabajo presentamos MuBPEL, la herramienta de mutación firme que implementa los operadores de mutación de [4] y permite comparar los mutantes generados con las composiciones originales. La herramienta es de código abierto (publicada bajo la Apache Software License 2.0) y se encuentra disponible a través de su web oficial [9].

En la siguiente sección estudiaremos la utilidad de la mutación firme en WS-BPEL 2.0, y a continuación describiremos las funcionalidades y el diseño de alto nivel de MuBPEL. Finalmente, cerraremos este texto resumiendo el estado actual de la herramienta y listando las líneas de trabajo actualmente abiertas.

2. Mutación firme en WS-BPEL 2.0

La mutación *fuerte* usada tradicionalmente compara la salida final del programa original con la del mutante [3]. Woodward y Halewood [10] introducen dos variantes, la mutación *débil* que compara el estado del programa justo después de haberse ejecutado la mutación, y la mutación *firme* que hace la comparación en algún punto posterior a la mutación. La mutación firme puede considerarse como un compromiso entre la mutación fuerte y la débil y subsume a ambas.

Un problema común al aplicar la mutación firme es que no está claro cómo seleccionar el punto en el que se realiza la comparación. Jackson y Woodward [5] consideran que Java y, en general, los lenguajes orientados a objetos son adecuados para la mutación firme, considerando a la salida de cada método ejecutado como un estado intermedio. Sin embargo, esta técnica sigue sin ser muy usada.

En este contexto, defendemos que la técnica de la mutación firme es ideal en un lenguaje de composición de servicios Web como WS-BPEL 2.0: los estados intermedios se encuentran bien definidos y son necesarios para las comparaciones. Esto se debe a que los efectos visibles de una composición web no consisten únicamente en su "salida final" (las respuestas a las peticiones de los clientes), sino también en toda la información enviada a los servicios que integra. Cada mensaje enviado a un servicio externo constituye una instantánea de un subconjunto bien definido del estado interno de la composición en ese momento, y es necesario saber si alguno de esos mensajes ha cambiado.

En la figura 1 podemos ver un ejemplo de las peticiones y respuestas intercambiadas entre una composición WS-BPEL que reúne a un servicio aprobador

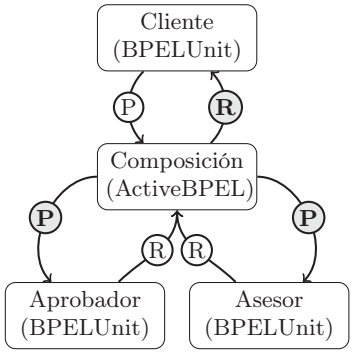


Figura 1. Peticiones (P) y respuestas (R) en una composición WS-BPEL: ejemplo del préstamo. Los mensajes utilizados en la mutación firme se destacan en negrita con fondo sombreado.

de préstamos y a un servicio asesor de riesgo de préstamos, los servicios integrados y el cliente. En esta composición, no sólo es importante la respuesta final («aprobado» o «no aprobado») sino también la información que le ha proporcionado al servicio asesor y al servicio aprobador a través de sus peticiones. Por lo tanto, al comparar la composición original y las mutantes se deben tener en cuenta los mensajes destacados en negrita.

3. MuBPEL

MuBPEL es una herramienta de línea de órdenes escrita en Java y publicada como código abierto en [9]. Su proceso de ejecución se divide en varias fases:

1. Analizador: detecta dónde se pueden aplicar los operadores de mutación.
2. Generador de mutantes: realiza las mutaciones en sí.
3. Sistema de ejecución: ejecuta un conjunto de casos de prueba para la herramienta BPELUnit [7] sobre la composición original y los mutantes en el motor ActiveBPEL [1] y compara los resultados según mutación firme.

MuBPEL incorpora un amplio abanico de opciones, pudiendo comparar composiciones ejecutándolas o usando sus salidas guardadas, parar tras hallar el primer caso de prueba que mata a cada mutante o ejecutar todos los casos de prueba o repartir el trabajo entre varios procesadores. ActiveBPEL es iniciado de forma transparente, al ejecutarse en un servidor Jetty embebido en MuBPEL.

Los resultados de la comparación se representan a través de una matriz M de tamaño $m \times t$, donde cada fila es uno de los m mutantes y cada columna es uno de los t casos de prueba. M_{ij} es 0 si el mutante i produce los mismos estados intermedios que el programa original para el caso de prueba j, 1 si produce estados distintos, y 2 si no pudo ejecutarse al no ser una composición válida.

4. Conclusiones y trabajo futuro

En este trabajo hemos presentado MuBPEL, nuestra herramienta de mutación firme de código abierto para WS-BPEL 2.0. Hemos propuesto el uso de mutación firme para los lenguajes de composición de servicios Web como WS-BPEL 2.0, debido a que los mensajes salientes de la composición constituyen un conjunto bien definido de estados intermedios a comparar. En los lenguajes imperativos tradicionales es más difícil definir este conjunto.

MuBPEL es una herramienta estable: nuestros trabajos actuales se centran en la corrección de defectos y en la mejora de los operadores de mutación. Como trabajo futuro planteamos incorporar una optimización del tiempo de ejecución y añadir la posibilidad de que la herramienta trabaje con ficheros de casos de prueba basados en plantillas.

Agradecimientos Este trabajo fue financiado por la beca de investigación PU-EPIF-FPI-C 2010-065 de la Universidad de Cádiz, por el proyecto MoDSOA (TIN2011-27242) del Programa Nacional de Investigación, Desarrollo e Innovación del Ministerio de Ciencia e Innovación y por el proyecto PR2011-004 del Plan de Promoción de la Investigación de la Universidad de Cádiz.

Referencias

1. ActiveVOS: Activebpel. `http://sourceforge.net/projects/activebpel502/`
2. Agrawal, H., Demillo, R., Hathaway, R., Hsu, W., Hsu, W., Krauser, E., Martin, R.J., Mathur, A., Spafford, E.: Design of mutant operators for the C programming language. Tech. Rep. SERC-TR-41-P, Software Engineering Research Center, Department of Computer Science, Purdue University, Indiana (1989)
3. DeMillo, R.A., Lipton, R.J., Sayward, F.G.: Hints on test data selection: Help for the practicing programmer. Computer 11(4), 34–41 (1978)
4. Estero-Botaro, A., Palomo-Lozano, F., Medina-Bulo, I.: Mutation operators for WS-BPEL 2.0. In: ICSSEA 2008, 21th International Conference on Software & Systems Engineering and their Applications (2008)
5. Jackson, D., Woodward, M.R.: Mutation Testing for the New Century, chap. Parallel firm mutation of Java programs, pp. 55–61. Kluwer Academic Publishers (2001)
6. N. King, K., Offutt, A.J.: A FORTRAN language system for mutation-based software testing. Software - Practice and Experience 21(7), 685–718 (1991)
7. Mayer, P., Lübke, D.: Towards a BPEL unit testing framework. In: TAV-WEB'06: Proceedings of the 2006 workshop on Testing, Analysis, and Verification of Web Services and Applications. pp. 33–42. ACM (2006)
8. OASIS: Web Services Business Process Execution Language 2.0. `http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html` (2007)
9. UCASE Research Group: MuBPEL. `https://neptuno.uca.es/redmine/projects/sources-fm/wiki/MuBPEL` (2012)
10. Woodward, M.R., Halewood, K.: From weak to strong, dead or alive? An analysis of some mutation testing issues. In: TVA'88: 2nd Workshop on Software Testing, Verification, and Analysis. pp. 152–158. IEEE Computer Society (1988)

Tutorial de Pruebas de Rendimiento

Federico Toledo Rodríguez[1] , Beatriz Pérez Lamancha[2], Macario Polo Usaola[3],

[1] Abstracta, Montevideo, Uruguay.
ftoledo@abstracta.com.uy
[2]Centro de Ensayos de Software, Universidad de la República, Montevideo, Uruguay
bperez@fing.edu.uy
[3]Universidad de Castilla-La Mancha, Ciudad Real, España
macario.polo@uclm.es

Abstract. Una prueba de desempeño (performance test) se define como una investigación técnica para determinar o validar la velocidad, escalabilidad y/o características de estabilidad de un sistema bajo prueba. Las pruebas de carga (load test) tienen como objetivo simular la realidad a la cual estará sometido el sistema en producción (lo cual se conoce como escenario) para analizar su desempeño ante esa situación. En este tutorial se presenta una metodología que se encuentra en el marco de la Ingeniería de Software más precisamente en el área de Verificación de Software, útil para realizar pruebas de carga, aunque puede ser extendida a otros tipos de pruebas de desempeño como por ejemplo pruebas de estrés o pruebas de picos. Uno de los desafíos de las pruebas de performance es lograr reducir riesgos del negocio y obtener información útil del sistema en tiempos reducidos, tratando de obtener información de valor que permita maximizar la relación costo/beneficio de la prueba.
El tutorial se orientaría principalmente a las áreas de desarrollo de sistemas empresariales, pruebas de software, metodologías y procesos.
El objetivo del curso es transmitir a los participantes la necesidad de la realización de este tipo de pruebas en forma oportuna, y una metodología clara como para poder llevar a cabo estas tareas, viendo con ejemplos y demostraciones las herramientas necesarias para la simulación de la carga de usuarios concurrentes (en particular OpenSTA), así como la monitorización de sistemas (a nivel de Sistema Operativo y de aplicación), introduciéndonos en el análisis de resultados.

Palabras claves: performance testing, metodología, aseguramiento de la calidad.

1 Motivación, objetivos, relevancia y audiencia potencial

El tutorial tiene un enfoque práctico, mostrando la aplicación de cada concepto teórico presentado. Vale destacar que en la industria se nota un aumento importante en la necesidad de realizar este tipo de pruebas, por lo que es de gran importancia tener presentes las mejores formas de realizar estos análisis, para dar resultados de

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 419-421, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

valor efectivos. Está basado en la experiencia de 7 años de trabajo en diferentes plataformas, diferentes herramientas para generación de carga y para monitorización. Todo lo que se imparta ha sido validado en la industria. El tutorial está enriquecido experiencias reales. Como se plantea el uso de herramientas *open source*, cualquiera de los participantes puede luego poner en práctica sin ningún costo extra todo lo que aprenda. Este tutorial está dirigido principalmente al personal de desarrollo, personal técnico y de especialidad en calidad de software.

2 Esquema de contenidos

1. Introducción
2. Visión general
    a. Tipos de Testing
    b. Rentabilidad de las pruebas de performance
3. Etapas de un test de performance
    a. Definición de las pruebas a realizar
    b. Automatización de las pruebas
        i. Herramientas disponibles en el mercado
    c. Armado del ambiente de prueba
    d. Ejecución de las pruebas
4. Gestión de un proyecto de testing de performance

3 Nombre, afiliación y breve biografía de los ponentes:

Federico Toledo Rodríguez (ponente)

Ingeniero en Computación egresado de la Universidad de la República en Uruguay, actualmente estudiando doctorado en informática en la Escuela Superior de Informática de la UCLM en Ciudad Real, incorporándose así al grupo de investigación Alarcos. Con más de 7 años de experiencia en consultoría, investigación y desarrollo vinculado al área de testing. Participación desde el año 2005 en proyectos de pruebas de rendimiento, tanto como analista, tester y líder de proyectos. Desde el año 2008 es uno de los cofundadores de Abstracta, la cual se dedica al desarrollo de herramientas que ayudan a simplificar las pruebas de software, y en particular a desarrollar GXtest, la cual es una herramienta de testing basado en modelos específica para automatizar pruebas en ambientes GeneXus.

Beatriz Pérez Lamancha (miembro del proyecto)

Profesor (grado 3) del Instituto de Computación de la Facultad de Ingeniería de la Universidad de la República. Trabajó desde la fundación del Centro de Ensayos de Software de Uruguay como Líder de Proyecto y Responsable de Capacitación. Obtuvo su doctorado en la Universidad de Castilla- La Mancha, dirigida por el Dr. Macario Polo y el Dr. Mario Piattini en el tema de Testing en Líneas de Producto de Software. Su línea de investigación principal es el testing, en especial proceso de testing, testing basado en modelos, testing combinatorio y automatización del testing.

Macario Polo Usaola (miembro del proyecto)

Profesor Titular de Lenguajes y Sistemas Informáticos en la Universidad de Castilla-La Mancha, en donde imparte las asignaturas Ingeniería del Software en el grado, Pruebas y Seguridad de Sistemas de Información en el máster investigador. Sus líneas de investigación están relacionadas con la automatización de actividades y tareas del ciclo de vida software, especialmente de las pruebas. Ha publicado, además, varias novelas, la última de las cuales es El pecador mudo.

A. Ruíz, L. Iribarne (Eds.): Actas de las "*XVII Jornadas de Ingeniería del Software y Bases de Datos (JISBD'2012)*", Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

Sesión Temática 5

Desarrollo de Software Dirigido por Modelos

Coordinadores: *Dr. Antonio Vallecillo y Dr. José Raul Romero*

Sesión Temática 5: *Desarrollo de Software Dirigido por Modelos*
Coordinadores: Dr. Antonio Vallecillo y Dr. José Raul Romero

Javier Luis Canovas Izquierdo and Jordi Cabot. *Creación Colaborativa de Lenguajes Específicos de Dominio*. (Emergente)

Javier Troya y Antonio Vallecillo. *On the Modular Specification of Non-Functional Properties in DSVLs*. (Emergente)

Alfonso Rodriguez, et al. *Secure Business Process model specification through a UML 2.0 Activity Diagram profile*. (Relevante)

Feliu Trias, et al. *Definición del dominio de las aplicaciones Web basadas en CMS: un Metamodelo Común para CMS*. (Regular)

María Gómez, et al. *MOSKitt4SPL: Tool support for Developing Self-Adaptive Systems*. (Herramienta)

Alvaro Jimenez, et al. *Aplicando los principios del DSDM al desarrollo de transformaciones de modelos en ETL*. (Regular)

Encarna Sosa Sánchez, et al. *Un proceso de modernización dirigido por modelos de sistemas web heredados hacia SOAs*. (Emergente)

Francisco Javier Bermúdez Ruiz and Jesús Joaquín García Molina. *Un framework basado en modelos para la modernización de datos*. (Regular)

Iván Santiago, et al. *iTrace: un framework para soportar el análisis de información de trazabilidad en proyectos de Desarrollo Software Dirigidos por Modelos*. (Regular)

Victor Manuel Bolinches Marin and José Angel Carsí Cubel. *Diseño de niveles y uso de motores en el desarrollo de videojuegos dirigido por modelos*. (Regular)

Pedro Sánchez, et al. *Introducing Safety Requirements Traceability Support in Model-Driven Development of Robotic Applications*. (Relevante)

Javier Espinazo Pagán, Jesús Sánchez Cuadrado and Jesús García Molina. *Un repositorio NoSQL para acceso escalable a modelos*. (Regular)

Ricardo Perez, et al. *A Family of Case Studies on Business Process Mining*. (Relevante)

Maria Gomez, Joan Fons and Vicente Pelechano. *Evolución de Sistemas Auto-Adaptables mediante Modelos en Tiempo de Ejecución*. (Regular)

Jesús Sánchez Cuadrado, et al. *Parametrización de las transformaciones horizontales en el modelo de herradura*. (Emergente)

Jesús Sánchez Cuadrado. *Transformación de modelos con Eclectic*. (Herramienta)

Manuel Wimmer, Loli Burgueño and Antonio Vallecillo. *Prueba de Transformaciones de Modelos con TractsTool*. (Herramienta)

Rober Morales-Chaparro, Juan Carlos Preciado and Fernando Sanchez-Figueroa. *Desarrollo dirigido por modelos de visualización de datos para la Web*. (Regular)

Pedro J. Clemente, et al. *Managing crosscutting concerns in component based systems using a model driven development approach*. (Relevante)

Creación Colaborativa de Lenguajes Específicos de Dominio

Javier Luis Cánovas Izquierdo, Jordi Cabot

AtlanMod, École des Mines de Nantes – INRIA – LINA, Nantes, France,
{javier.canovas,jordi.cabot}@inria.fr

Resumen El desarrollo de sofware es un proceso donde participan muchos actores, principalmente los desarrolladores y los clientes del producto. En la actualidad, procesos de desarrollo como los basados en metodologías ágiles proponen la participación de forma directa de los usuarios o clientes. La idea clave es definir procesos guiados por la comunidad donde todos los participantes (técnicos y no técnicos) colaboran para que el producto satisfaga los requisitos.
Esta aproximación es especialmente interesante en el ámbito del desarrollo de lenguajes específicos de dominio (DSL). Sin embargo, aunque estos lenguajes están destinados a una comunidad de usuarios expertos de un dominio concreto, actualmente dichos usuarios tienen poca (o nula) participación en el desarrollo. Nuestra propuesta consiste en incorporar el aspecto colaborativo en los procesos de desarrollo de DSLs, permitiendo a la comunidad de usuarios del lenguaje participar activamente en su creación y evolución. Para ello proponemos adaptar *Collaboro*, un lenguaje para representar las actividades de colaboración que surgen durante el desarrollo de DSLs, para ser utilizado a lo largo de todo el proceso.

1. Introducción

En los últimos años la mayoría de los esfuerzos para hacer más eficientes los procesos de desarrollo software se han centrado fundamentalmente en un determinado actor: los desarrolladores. El usuario final (o cliente) participa durante la elicitación de requisitos pero es ignorado hasta la etapa de pruebas, provocando normalmente que las aplicaciones no satisfagan las espectativas [1]. En respuesta a esta situación, diferentes metodologías de desarrollo, como las llamadas ágiles [2], y otros trabajos como [1,3] proponen la colaboración de los usuarios finales durante todo el proceso, permitiendo una validación continua.

El desarrollo de aplicaciones libres y de código abierto (*Free and Open Source Software*, FOSS) son el principal ejemplo de desarrollos colaborativos. Estos procesos utilizan sistemas para promover la participación como, por ejemplo, sistemas de votación (como en Apache [4]) o asignación de tareas (como en Mozilla [5]). Otro ejemplo representativo de sistemas colaborativos se puede encontrar en la web social, como los sitios web de *stackExchange*[1], donde los usuarios de la comunidad colaboran para la resolución de problemas de otros usuarios.

La participación de los usuarios finales cobra especial importancia en el desarrollo específico de dominio, en particular, en los lenguajes específicos de dominio (*Domain*

[1] http://www.stackexchange.com

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 425-430, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

*Specific Language*s, DSLs). En los DSLs, la colaboración de los usuarios se hace imprescindible ya que estos lenguajes están destinados a un dominio de aplicación del cual son precisamente expertos. Sin embargo, aunque existen un conjunto de técnicas y recomendaciones para el desarrollo de DSLs [6,7,8], la mayoría se centra en los artefactos y tareas a llevar a cabo por los desarrolladores.

Herramientas como Bugzilla[2] o Trac[3] permiten el desarrollo colaborativo de sistemas software pero no están adaptadas para tratar con DSLs. Por otro lado en trabajos como [1,3] se comenta la utilidad de la participación de los usuarios en algunas fases que utilizan modelos durante el desarrollo de software, sin embargo, no presentan la colaboración como un proceso que ayude a guiar el desarrollo. Finalmente herramientas como [9,10] solo permiten la colaboración online, lo que limita la participación y no permite la representación de las interacciones surgidas.

Nuestra propuesta consiste en hacer colaborativo el proceso de desarrollo de DSLs, conviertiéndolo en uno más democrático que tenga en cuenta las opiniones y sugerencias de la comunidad (usuarios y desarrolladores) existente alrededor del lenguaje. De forma parecida al desarrollo de FOSS y a la web social, los miembros de la comunidad pueden proponer, discutir y decidir qué cambios y soluciones son las más adecuadas. En este sentido, la colaboración se refiere a la puesta en común de propuestas y toma de decisiones conjunta. Para ello definimos un DSL, denominado Collaboro, para representar las colaboraciones de los miembros de la comunidad, es decir, las propuestas de cambio, soluciones y comentarios. Estas colaboraciones permiten llegar a acuerdos acerca de los cambios a incorporar al lenguaje así como mantener un registro de dichos cambios, permitiendo guiar el proceso de desarrollo y conocer qué motivó la creación de cada elemento, respectivamente. En [11] presentamos una primera versión del lenguaje para el desarrollo colaborativo de la sintaxis abstracta de los DSLs. En este trabajo exploramos su adaptación para el diseño de la sintaxis concreta así como otras mejoras que permitan explotar el potencial de la comunidad a lo largo de todo el proceso.

El resto del artículo está organizado de la siguiente manera. La Sección 2 contrasta un proceso de desarrollo de DSL tradicional con uno colaborativo. La Sección 3 describe Collaboro y, finalmente, la Sección 4 presenta las líneas de trabajo principales.

2. Modelo tradicional vs. colaborativo

Un DSL está compuesto de tres elementos principales [12]: sintaxis abstracta, sintaxis concreta y semántica. La sintaxis abstracta define los conceptos principales del lenguaje y sus relaciones, así como un conjunto de restricciones. La sintaxis concreta define la notación del lenguaje, que puede ser textual, gráfica o híbrida. Finalmente, la semántica del DSL se define generalmente utilizando una aproximación traslacional.

Según [6], el proceso de creación de un DSL está compuesto de cinco fases: decision, análisis, diseño, implementación y despliegue. En la fase de decisión se identifica la necesidad de crear un DSL para un dominio particular. En la fase de análisis se estudia el dominio de aplicación para, a continuación, crear el lenguaje en las fases de diseño e implementación. El lenguaje está listo para ser utilizado en la fase de despliegue.

[2] http://www.bugzilla.org
[3] http://trac.edgewall.org

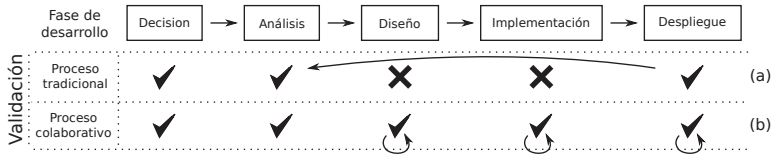


Figura 1. Validación en (a) un proceso tradicional y en (b) un proceso colaborativo.

En el modelo tradicional de desarrollo, los expertos del dominio participan en las primeras dos fases y no vuelven a colaborar hasta la última fase, donde pueden comprobar si el lenguaje cumple con sus espectativas. Así, la validación del lenguaje se retrasa hasta el final del proceso, donde pueden aparecer problemas derivados de una incorrecta interpretación de los requisitos. Esta situación normalmente implica un reinicio del proceso para resolver los problemas identificados, provocando un incremento en el coste y esfuerzo de creación del lenguaje. La Figura 1a muestra las fases del proceso e indica aquellas en las que los usuarios participan. Además, también ilustra cómo los problemas indentificados en la última fase fuerzan a un reinicio del proceso.

En un proceso en el que participa la comunidad, tanto los desarrolladores como los usuarios finales del lenguaje pueden colaborar activamente en todas las fases del desarrollo, particularmente las enfocadas al diseño e implementación. De esta forma, los usuarios ya no tienen que esperar al final del proceso para validar el lenguaje sino que cada fase es validada conforme se va realizando (ver Figura 1b). Por ejemplo, en un lenguaje para representar rutas en una empresa de transportes, una vez la comunidad de usuarios decide satisfactoriamente crear el lenguaje y se supera la fase de análisis, se procede a su diseño e implementación. En una primera versión del lenguaje, la sintaxis abstracta incluye los conceptos de *vehículo* y *ruta*, compuesta de *puntos de parada*. Por falta de espacio y dada la simplicidad del lenguaje, no mostramos el metamodelo.

En el modelo tradicional, una vez definida la sintaxis abstracta, el proceso de implementación continuaría, definiendo la sintaxis concreta del lenguaje y el resto de herramientas. Por ejemplo, la sintaxis concreta podría consistir en una representación gráfica de las rutas, puntos de parada y camiones sobre un mapa geográfico. Los usuarios finales deberían esperar hasta la fase de despliege para poder validar y detectar posibles errores en el lenguaje. Por ejemplo, los usuarios podrían considerar que una característica crucial es representar el estado de una ruta en cuestión del tráfico para poder calcular mejor qué dirección tomar. De esta forma, la ruta debería mostrarse en diferentes colores, representado el estado del tráfico. Debido a que esta característica no fue considerada en la sintaxis abstracta del lenguaje, ésta y el resto de componentes del lenguaje, como la sintaxis concreta, deberían ser actualizados para contemplar el cambio.

En un proceso colaborativo, la falta de esta característica puede ser informada por la comunidad durante el mismo proceso de definición de la sintaxis abstracta. Además, la comunidad no solo puede detectar el problema sino también proponer soluciones y decidir en conjunto cuál debería ser incorporada. Esta tarea se ve facilitada por el hecho de trabajar con un DSL de un dominio donde la comunidad es experta, permitiendo a los miembros de ésta colaborar activamente. Por ejemplo, un usuario podría informar de la necesidad de la nueva característica y, a continuación, un desarrollador podría proponer una solución para adaptar el lenguaje, la cual sería discutida, valorada y votada por

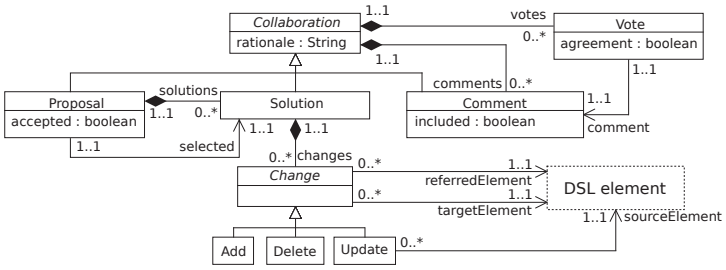


Figura 2. Parte principal del metamodelo de sintaxis abstracta de Collaboro.

la comunidad para su posterior implementación. Permitir este grado de participación requeriría poder representar las colaboraciones que surgen, lo cual permitiría además llevar un registro de todos los cambios introducidos.

3. Collaboro

Para hacer colaborativo un proceso tradicional de desarrollo de DSLs definimos Collaboro [11], un lenguaje que permite representar explícitamente las colaboraciones que surgen en la comunidad durante el proceso. Estas colaboraciones son expresadas como modelos y permiten llevar un control de los cambios aplicados durante el desarrollo del lenguaje. Collaboro, creado colaborativamente en nuestro grupo de investigación, ha sido desarrollado como un plugin de Eclipse y puede ser descargado desde su website[4].

La parte principal del metamodelo de sintaxis abstracta de Collaboro se muestra en la Figura 2 ([11] contiene una descripción más detallada). Este metamodelo permite representar información estática y dinámica. La parte estática de Collaboro representa las colaboraciones, que pueden ser: propuestas de cambio (metaclase `Proposal`), propuestas de solución (metaclase `Solution`) y comentarios (metaclase `Comment`). Cada elemento incluye una explicación en lenguaje natural (atributo `rationale`).

Las propuestas de solución describen los cambios a realizar, que pueden ser: añadir (metaclase `Add`), borrar (metaclase `Delete`) o actualizar (metaclase `Update`). Los cambios indican el elemento a modificar (referencia `referredElement`) y el elemento a cambiar (referencia `target`). En el caso de `Update`, también se indica el elemento previo al cambio (referencia `source`). Los elementos involucrados en estas referencias difieren dependiendo de la parte del DSL a cambiar, por ejemplo, pueden referir a las metaclases de la sintaxis abstracta del lenguaje que se está desarrollando.

Por otro lado, la parte dinámica registra los procesos de decisión para seleccionar aquellas propuestas de cambio y soluciones que deben incorporarse al lenguaje. Un usuario puede votar (metaclase `Vote`) a favor o en contra de una colaboración. Según los votos recibidos, una propuesta puede ser aceptada (atributo `accepted`), una solución seleccionada (atributo `selected`) o un comentario incluido (atributo `included`) para alterar la propuesta/solución comentada. El proceso de decisión puede basarse en diferentes algoritmos, por ejemplo, acuerdo por mayoría simple o absoluta.

[4] http://code.google.com/a/eclipselabs.org/p/collaboro

4. Plan de trabajo. Extensión de Collaboro

Nuestro objetivo es explorar la adaptación de Collaboro para cubrir otros elementos del desarrollo de DSLs así como estudiar otras mejoras que exploten el potencial de la comunidad. A continuación describimos las líneas de trabajo identificadas.

Sintaxis concreta. Dado el número de decisiones de diseño a tratar (gráfica y/o textual, herramientas, etc), la sintaxis concreta es una de las partes más complicadas de abordar colaborativamente. Una solución para facilitar su desarrollo colaborativo sería la definición de un metamodelo básico para sintaxis textuales y gráficas, que definiría, por ejemplo, elementos como *keyword* o *figura*, respectivamente. Los cambios que impliquen modificar la sintaxis concreta se podrían representar con Collaboro (adaptando las referencias de la jerarquía `ModelChange`), permitiendo a los usuarios de la comunidad discutir a alto nivel cómo desean que se represente el lenguaje.

Definición de cambios mediante ejemplos. Actualmente los cambios a aplicar se definen según los elementos del lenguaje. Una alternativa que facilitaría esta labor sería permitir especificar los cambios a través de ejemplos del lenguaje (posiblemente inconsistentes con la última versión de éste). De esta forma, los usuarios podrían expresar qué quieren poder definir con el lenguaje. Para soportar esta extensión, nuestra aproximación debería ser capaz de derivar modelos Colloboro a partir de los ejemplos.

Visualización de los cambios. Para facilitar la comprensión de las propuestas de solución así como ayudar a la decisión de su aceptación, sería interesante disponer de un generador de ejemplos de modelos. De esta forma, cualquier usuario de la comunidad podría visualizar cómo afectarían los cambios propuestos en la solución.

Políticas de decisión. La implementación de diferentes políticas de decisión permitiría adaptar la colaboración a la comunidad, por ejemplo, ponderando los votos de acuerdo a qué usuario está votando. Por otro lado, creemos que la formalización de las discusiones asociadas a los cambios, representadas actualmente en lenguaje natural, podrían enriquecer los procesos de decisión. Finalmente, el análisis de las decisiones adoptadas serviría de fuente de información para identificar patrones de calidad en lenguajes.

Extensión de soporte de la sintaxis abstracta. Aunque el desarrollo de la sintaxis abstracta ya ha sido abordado en la primera implementación de Collaboro, todavía no se ha abordado la definición de sus restricciones. Esta definición se realiza normalmente por medio de un lenguaje declarativo como OCL, lo cual normalmente requiere de conocimientos avanzados. Dado que las expresiones OCL pueden representarse como modelos, nuestra primera propuesta consiste en adaptar Collaboro para permitir que los cambios descritos en las soluciones puedan tratar con dichos modelos (adaptando también la jerarquía `ModelChange`). De esta forma, los expertos en OCL de la comunidad integrarían las restricciones expresadas por usuarios. Más adelante nos gustaría estudiar la posibilidad definir estas restricciones con algún lenguaje que facilite la colaboración de todos los miembros de la comunidad.

Automatización de la generación del DSL. Con el soporte a la sintaxis concreta, los modelos Collaboro podrían especificar las relaciones entre las sintaxis abstracta y concreta, permitiendo generar la información necesaria por herramientas de definición de DSLs para la implementación del lenguaje. Estas herramientas normalmente requieren una definición de elementos de notación y una definición de correspondencias entre

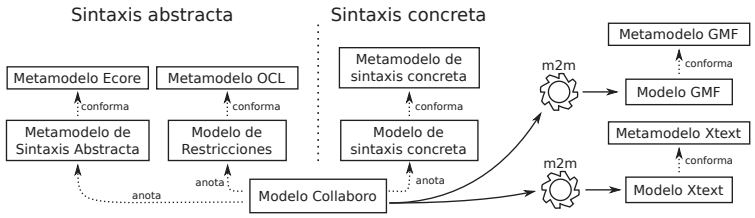


Figura 3. Aplicación de Collaboro para las sintaxis y generación de un DSL.

dichos elementos y los conceptos de la sintaxis abstracta. Dado que estas dos definiciones se representan generalmente como modelos (este es el caso de herramientas como XText[5] y GMF[6] para DSLs textuales y gráficos, respectivamente), los modelos Collaboro podrían ser transformados para obtener los modelos de la herramienta destino. La Figura 3 muestra el uso de Colloboro con los modelos involucrados en la definición colaborativa de las sintaxis concreta y abstracta así como las transformaciones de modelo requeridas, por ejemplo, para GMF y/o Xtext, para generar el DSL.

Sintaxis para Collaboro. Actualmente la manipulación de modelos Collaboro se realiza en el entorno Eclipse pero nos gustaría explorar otras alternativas para facilitar el acceso a los usuarios, por ejemplo, con una sintaxis textual o integrándolo en la web.

Referencias

1. Hildenbrand, T., Rothlauf, F., Geisser, M., Heinzl, A., Kude, T.: Approaches to collaborative software development. In: CISIS Conference, IEEE (2008) 523–528
2. Agile Manifesto. http://agilemanifesto.org
3. Whitehead, J.: Collaboration in software engineering: A roadmap. In: FOSE Conference, IEEE (2007) 214–225
4. Fielding, R.T.: Shared leadership in the apache project. Commun. ACM **42** (1999) 42–43
5. Mockus, A., Fielding, R.T., Herbsleb, J.D.: Two case studies of open source software development: Apache and mozilla. ACM Trans. Softw. Eng. Methodol. **11** (2002) 309–346
6. Mernik, M., Heering, J., Sloane, A.M.: When and how to develop domain-specific languages. ACM Comput. Surv. **37** (2005) 316–344
7. Kelly, S., Pohjonen, R.: Worst practices for domain-specific modeling. IEEE Software **26**(4) (2009) 22–29
8. Völter, M.: MD*/DSL best practices. http://voelter.de/data/pub/dslbestpractices-2011update.pdf
9. Brosch, P., Seidl, M., Wieland, K., Wimmer, M., Langer, P.: We can work it out: Collaborative conflict resolution in model versioning. In: ECSCW, Springer (2009) 207–214
10. Gallardo, J., Bravo, C., Redondo, M.A.: A model-driven development method for collaborative modeling tools. Network and Computer Applications **35** (2012) 1086–1105
11. Cánovas Izquierdo, J.L., Cabot, J.: Community-driven language development. In: MiSE Workshop, IEEE (2012) 29–35
12. Kleppe, A.: Software Language Engineering: Creating Domain-Specific Languages Using Metamodels. Addison Wesley (2008)

[5] http://www.eclipse.org/Xtext
[6] http://www.eclipse.org/gmf

On the Modular Specification of Non-Functional Properties in DSVLs

Javier Troya, Antonio Vallecillo, and Francisco Durán

GISUM/Atenea Research Group. Universidad de Málaga (Spain)
{javiertc,av,duran}@lcc.uma.es

Abstract. In previous work we have presented an approach to monitor non-functional properties of systems modeled in terms of domain specific visual languages using *observers*. In this work we present an approach to decouple the definition of observers behavior and systems behavior. Having a library with different kinds of observers behavior, and having the behavioral definition of the system, weaving links can be established among them in order to include observers in the system behavioral specification.

Key words: DSVLs, weaving mechanisms, observers

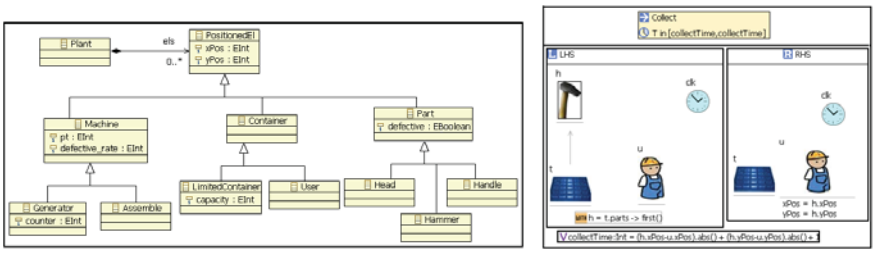
1 Introduction

Domain specific visual languages (DSVLs) play a cornerstone role in Model-Driven Engineering (MDE) for representing models and metamodels. The benefits of using DSVLs is that they provide an intuitive notation, closer to the language of the domain expert, and at the right level of abstraction. The Software Engineering community's efforts have been progressively evolving from the specification of the structural aspects of a system to modeling its behavioral dynamics. Thus, several proposals already exist for modeling the structure and behavior of a system. Some of these proposals also come with supporting environments for animating or executing the specifications, based on the transformations of the models into other models that can be executed [1, 2].

In previous work [3] we proposed our own approach to monitor non-functional properties of DSVLs. The idea is to integrate new objects, named *observers*, in the system specifications that capture such properties. Our proposal is based on the observation of the execution of the system actions and of the state of its constituent objects in the case of DSVLs that specify behavior in terms of rules. The use of observer objects enables the analysis of some of the properties usually pursued by simulation, like cycle-times, busy and idle cycles, mean-time between failures, throughput, delay, etc. The OMG, in turn, defines different kinds of observers in the MARTE specification [4], which are similar to ours. However, they cannot be used to describe requirements and constraints on models, as we do. Furthermore, we can use our observers to dynamically change the system behavior, in contrast with the more "static" nature of MARTE observers.

In our approach, users define their own observers, according to the non-functional properties they want to monitor. Then, these observers are to be included in the behavioral rules of systems. The resulting rules are difficult to maintain, since the observers

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 431-436, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.



(a) Metamodel. (b) Behavioral Rule.

Fig. 1. Production Line System

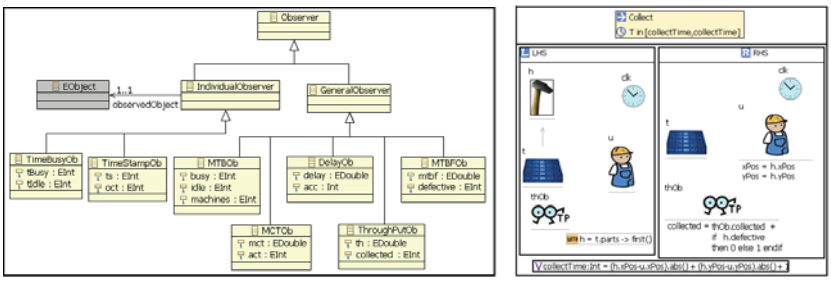
and systems specifications are mixed. In this paper we propose an approach to decouple the process of integrating observers in the systems specification. Having a library where different kinds of observers generic behaviors are available, the idea is to weave these behaviors with the systems behavioral rules. In this sense, our approach may seem similar to the Software Metrics Meta-model (SMM) [5] defined by the OMG, a metamodel for representing measurement information related to software, its operation and design. However, our approach is more flexible since new metamodels and metrics to monitor non-functional properties can be defined by the modeler. Furthermore, we give semantics for defining the dynamic behavior of systems and their non-functional properties.

After this introduction, Section 2 briefly describes our current approach for monitoring non-functional properties of systems. Then, in Section 3 we present our ideas for a modular specification of observers. Finally, Section 4 concludes the paper and outlines how we would like to continue this work.

2 Monitoring Non-Functional Properties of Systems using Observers

In this section we briefly describe our current approach for the specification and monitoring of non-functional properties of systems in *e-Motions* [6].

The first step is to define our DSVL. DSVLs are defined in terms of three main elements: abstract syntax, concrete syntax and semantics. The abstract syntax defines the domain concepts that the language is able to represent, and is defined by a metamodel. The concrete syntax defines the notation of the language, and in *e-Motions* it is defined by assigning an icon to each concept in the metamodel. The semantics describe the meaning of the models represented in the language, and in case of models of dynamic systems (such as ours) the semantics of a model describe the effects of executing that model. In our case, semantics are specified by a set of behavioral rules. Snapshots of a part of a metamodel and a behavioral rule for a particular example of a production line system are shown in Figure 1. Regarding the metamodel, Machines generate and assemble Parts and Containers either transport or keep these Parts. The rule models how



(a) Observers Metamodel.

(b) Behavioral Rule with Observers.

Fig. 2. Observers for the Production Line System

a Hammer that is placed in a LimitedContainer is collected by a User. The complete description of this case study is presented in [7].

Once the system behavior has been specified, it is time to add observers to the rules. The first step is to identify which non-functional properties the user wants to monitor: throughput, mean-time between failures, delays, response times, etc. Observers are specified by means of a metamodel (an observers metamodel for the production line system is shown in Figure 2(a)), which is then combined with the system metamodel to be able to use the observers in the DSVL. In the observers metamodel it can be seen that we define observers of two types: *Individual* and *General*. The former are those which are attached to individual objects to monitor their state and/or behavior, while the latter monitor individual observers, as well as the remaining objects in the system, to build derived measures for the non-functional properties we want to monitor. The next step is to include observers in the behavioral rules in order to monitor non-functional properties of our DSVL. The observer added in Figure 2(b) is meant to monitor the throughput of the production line. Concretely, in this rule the observer updates its collected attribute everytime a new Hammer is collected by the User. The specifications of systems with observers are then translated into the corresponding formal specifications in Maude [8]. In fact, since the Maude rewriting logic specifications are executable, they can be used as a prototype of the system, which allows us to simulate and analyze it. After the simulation, observers contain information about how the system behaved in terms of its non-functional properties.

In our current approach, the addition of observers may require to change the existing behavioral models to a large extent, rendering in some cases a fairly complex, difficult to understand and potentially hard to maintain system models. This is because the addition of observers in the behavioral rules depends directly on the type of system we are dealing with. In this way, different kinds of observers must be defined for each system. Next section presents an approach to overcome these limitations.

3 A Modular Approach for the Specification of Observers

In this section we propose the use of aspect-oriented techniques, whereby a modular specification of the behavior of observers is provided, and then woven with the system behavioral rules. Although this approach limits the flexibility required for observers in some situations, it can be used for the majority of properties. It also provides a modular approach to the specification and monitoring of individual properties, for which observers can be independently defined and reused across system specifications. Thus, the idea is to define a library with different observers behavior, which can then be reused by concrete systems to incorporate observers within their behavior specifications. This way, observers metamodel and rules are defined only once and used with different kinds of systems.

Our approach is based on standard software measurement approaches, which define *base* and *derived* measures [9]. The former allow measuring individual object attributes, while the latter build on the values of base measures to define aggregated metrics. Similarly, we have our *individual* and *general* observers. Although of different nature, both kinds of observers can be specified using a similar approach. Their behavior, once specified, can be woven to the functional behavior of a system to produce the complete system specifications. Thus, the behavior of observers follows a regular pattern, that corresponds to their life-cycle: creation, monitoring and termination. Rules are defined for each of these phases. Due to the space limitation, in this paper we describe the generic behavior for general observers. The generic definition of the behavior of all the observers shown in Figure 2(a) can be found at [10].

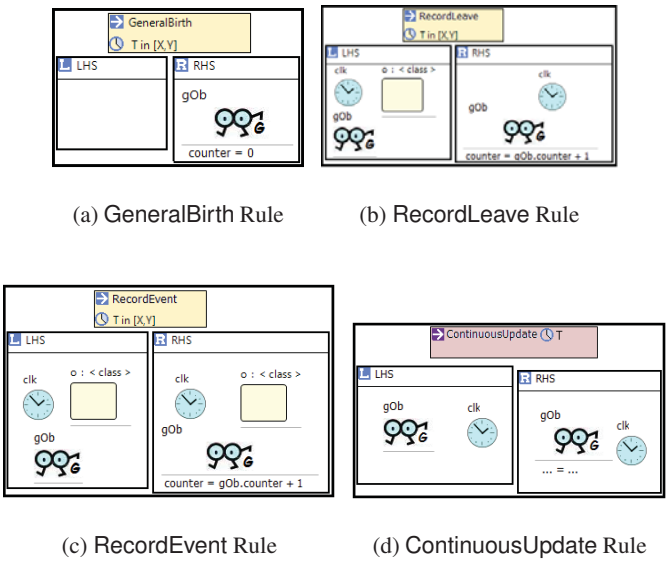
3.1 Generic Behavior of General Observers

The behavior of general observers is normally determined by four of rules. We show them here in general, and then these rules are specialized when defining the behavior of a concrete kind of observer (the behavior of all the observers of the production line system example can be found in [10]). The first rule (GeneralBirth, shown in Figure 3(a)) specifies the creation of a general observer. Since they are created at start time, this rule is normally woven to the initial rule of the system. Two rules specify how global observers update their state variables, depending on whether they do it when an object disappears from the system, or when the object participates in a rule. Thus, generic rule RecordLeave (Fig. 3(b)) shows how a counter attribute is updated when an object leaves the system. Similarly, rule RecordEvent (Fig. 3(c)) models the behavior of a global observer that records that an object has participated in a rule. Some attributes of some general observers need to be updated as time moves forward. Such a behavior is modeled with ongoing rules, whose generic form is shown in Fig. 3(d).

3.2 Weaving the Rules

Once we have the rules that specify the observers behavior independently from any concrete system specification, we need to weave them with the system rules. For that we use a weaving model that uses the AtlanMod Model Weaver (AWM[1]) to define the

[1] http://wiki.eclipse.org/AMW



(a) GeneralBirth Rule (b) RecordLeave Rule

(c) RecordEvent Rule (d) ContinuousUpdate Rule

Fig. 3. Generic rules for general observers.

correspondences between the *generic* objects in the observers rules and the concrete objects that appear in the system behavioral rules. In this case, the object in the generic rule and the one in the concrete rule that match will be woven, and the remaining elements in the generic rule will be copied to the concrete one. It is also possible to define a correspondence between one observer rule and one system rule. In this case, all the elements from the observer rule will be copied to the system rule. When defining the weaving links between the observers and the system rules it is possible to add expressions that overwrite how the values of the attributes are calculated in the observer rule. In this way we allow some kind of rule *parametrization*. Finally, it is possible (and very common) to establish correspondences between several observers rules and one system rule, when we want to add more than one kind of observer to a rule. Only one final rule is produced with the results of all weaves. To illustrate this approach, let us apply it to the production line system example. Starting from the behavioral rules without observers and assuming that we have defined the rules for the observers, in Figure 4 we present the matching for inserting the general ThroughPutOb observer in the Collect behavioral rule from Figure 1(b). The weaving link is defined between the Hammer object in the Collect rule of the system and the generic object in the RecordLeaveTP rule. The effect of this binding is to include the ThroughPutOb observer in both the LHS and RHS of the Collect rule, creating the rule shown in Fig. 2(b). Note that the expression used to calculate the value of the collected attribute in the RHS of the observer rule has been overwritten by a new expression (this is indicated in the figure inside the box between the two woven rules).

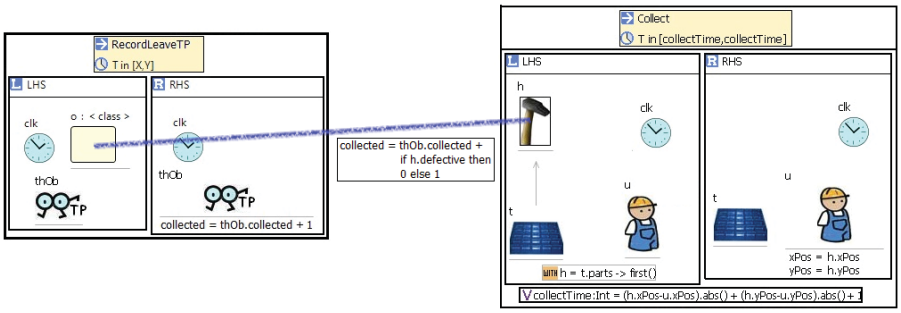


Fig. 4. Weaving the RecordLeaveTP and Collect rules.

4 Conclusions and Future Work

We have described our current approach to specify and monitor non-functional properties of DSVLs. We have also presented how we want to extend that approach in order to include observers in the systems behavioral rules in a decoupled and modular way. The idea is to have a library with the behavior of different kinds of observers, and then apply weaves between these rules and the systems rules. Our plan for future work is to continue the study of this approach in order to completely implement it.

References

1. Efroni, S., Harel, D., Cohen, I.R.: Reactive animation: Realistic modeling of complex dynamic systems. Computer **38**(1) (2005) 38–47
2. Ermel, C., Ehrig, H.: Behavior-preserving simulation-to-animation model and rule transformations. ENTCS **213**(1) (2008) 55–74
3. Troya, J., Rivera, J.E., Vallecillo, A.: Simulating domain specific visual models by observation. In: Proc. of the Symposium on Theory of Modeling and Simulation (DEVS'10), Orlando, FL (US) (2010)
4. OMG: A UML Profile for MARTE: Modeling and Analyzing Real-Time and Embedded Systems. OMG. (2008)
5. OMG: Software Metrics Meta-Model (SMM). OMG. (2009)
6. Rivera, J.E., Durán, F., Vallecillo, A.: A graphical approach for modeling time-dependent behavior of DSLs. In: Proc. of VL/HCC'09, Corvallis, Oregon (US) (2009)
7. Atenea: Non-functional monitoring in the PLS case study (2011) `http://atenea.lcc.uma.es/index.php/Main_Page/Resources/E-motions/PLSObExample`.
8. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.: All About Maude – A High-Performance Logical Framework. Number 4350. Springer, Heidelberg, Germany (2007)
9. García, F., Bertoa, M.F., Calero, C., Vallecillo, A., Ruíz, F., Piattini, M., Genero, M.: Towards a consistent terminology for software measurement. Information and Software Technology **48**(8) (2006) 631–644
10. Atenea: Modular Approach in the PLS case study (2012) `http://atenea.lcc.uma.es/index.php/Main_Page/Resources/E-motions/PLSObModExample`.

Secure Business Process model specification through a UML 2.0 Activity Diagram profile

Alfonso Rodríguez[1], Eduardo Fernández-Medina[2], Juan Trujillo[3], and Mario Piattini[2]

[1] Computer Science and Information Technology Department,
University of Bio-Bio, Chillán, Chile
alfonso@ubiobio.cl
[2] GSyA Research Group, Information Systems and Technologies Department,
University of Castilla-La Mancha, Ciudad Real, Spain
{Eduardo.FdezMedina, Mario.Piattini}@uclm.es
[3] LUCENTIA Research Group, Department of Software and Computing Systems,
University of Alicante, Alicante, Spain
jtrujillo@ dlsi.ua.es

1 Summary

Business processes are currently important resources both for enterprises' performance and to enable them to maintain their competitiveness. In recent years, the languages used for business process representation have been improved and new notations have appeared. In spite of the fact that the importance of business process security is accepted, the business analyst perspective in relation to security has so far scarcely been dealt with. Moreover, security requirements cannot be represented in modern business process modeling notations.

In this paper, we have presented an extension of the UML 2.0 Activity Diagram (UML 2.0-AD) which allows us to specify security requirements in the business process domain. This specification corresponds with a Computation Independent Model (CIM) within the MDA (Model Driven Architect) approach. We have based our proposal on UML 2.0-AD for three main reasons: (i) UML 2.0 description significantly improves the business process representation through Activity Diagrams, (ii) UML can easily be extended, thus allowing it to be tailored to a specific domain and (iii) UML modeling is dominant in the software industry and this eases the transformation of business process models into models which are closer to implementation.

In our proposal, called BPSec (Business Process Security), we use the approach driven by models, MDA, because it establishes that a business process corresponds to a Computation Independent Model, while UML artifacts such as analysis classes and use cases correspond to platform independent models. Thus, a model transformation from a Secure Business Process (SBP) model to analysis classes and use cases is the transformation from CIM to Platform Independent Model (PIM), according to the MDA paradigm.

In Figure 1, we show all the details of our proposal. We have colored the following elements in dark grey: (i) BPSec; the UML 2.0-AD extension presented in details in this work, (ii) M-BPSec, a method that we have designed for the ordered and

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 437-438, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

systematic construction of secure business processes models, (iii) BPSec-Tool, a prototype that supports M-BPSec application, (iv) the Secure Business Process model that is obtained from the application of the method supported by the tool and (v) a set of rules described using QVT (Query/View/Transformation), that have been incorporated into the method and the tool and which describes the transformation from CIM to PIM.

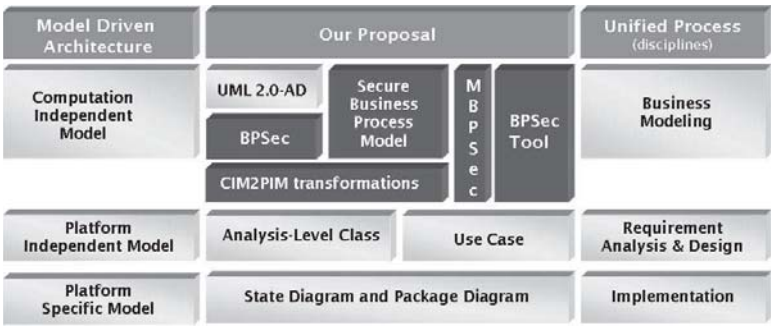


Fig. **1**: Our proposal overview

The CIM to PIM transformations shown in Figure 1 are not the main objective of this work and we will only present the results obtained from their application. These transformations permit us to obtain a set of UML artifacts (analysis classes and use cases diagrams) that correspond to PIM models from a Secure Business Process (SBP) description (in CIM).

The stages of the Unified Process (UP) are shown in the last column of Figure 1. Our purpose is to show that not only SBP specification but also analysis classes and use cases diagrams can be used in a complementary way in a consolidated and successful software development process such as the Unified Process. Thus, taking into consideration a certain parallelism between our proposal and the Unified Process, the SBP model will be built in the "Business Model" stage and the analysis classes and use cases will be defined and refined in the "Requirements" and "Analysis & Design" stages.

References

1. Rodríguez, A., Fernández-Medina, E., Trujillo J., and Piattini, M.: Secure Business Process Model specification through a UML 2.0 Activity Diagram Profile**.** Decision Support Systems, Volume 51, Issue 3, 446–465 (2011).

Definición del dominio de las aplicaciones Web basadas en CMS: un Metamodelo Común para CMS*

Feliu Trias, Valeria de Castro, Marcos López-Sanz, Esperanza Marcos

Kybele Research Group
Rey Juan Carlos University
Tulipán S/N, 28933, Móstoles, Madrid, Spain.
{feliu.trias, valeria.decastro, marcos.lopez, esperanza.marcos}@urjc.es

Abstract. En los últimos años, los Sistemas de Gestión de Contenidos (*Content Management System, CMS*) han aumentado su presencia en organizaciones y empresas gracias a las ventajas que ofrecen para la gestión del contenido digital. En concreto, las empresas han empezado a utilizar CMSs como plataforma de desarrollo para sus aplicaciones Web. Por esta razón, las aplicaciones Web basadas en CMS han ganado popularidad rápidamente. A pesar de ello, los métodos de ingeniería Web que existen en la actualidad no están del todo adaptados al dominio de los CMS. Esto queda reflejado en los lenguajes de modelado que proponen los métodos de ingeniería Web dirigidos por modelos ya que carecen de expresividad para representar y capturar los elementos necesarios para desarrollar este tipo de aplicaciones Web. Para contribuir a la solución de este problema presentamos en este artículo un metamodelo que recoge los principales conceptos para modelar aplicaciones Web basadas en CMS, CMS-CM (*CMS Common Metamodel*). Este metamodelo podría ser utilizado para extender los lenguajes de modelado ya existentes, además de servir de base a nuevos lenguajes de modelado específicos para el ámbito de los CMS.

Palabras clave: Ingeniería Web, Desarrollo Dirigido por Modelos, Lenguaje Específico de Dominio, Sistema de Gestión de Contenidos.

1 Introducción

En los últimos años, Internet se ha convertido en una plataforma que soporta sofisticadas aplicaciones empresariales y procesos de negocio complejos [1]. Hoy en día es en una plataforma imprescindible para la expansión del negocio de una empresa ya que le aporta ventajas competitivas, le permite una colaboración global y una integración con socios externos [2]. A consecuencia de esto, el número de aplicaciones Web desarrolladas por la industria ha aumentado de forma espectacular en la última década.

* Esta investigación ha sido financiada por el Proyecto MASAI (TIN-2011-22617) del Ministerio de Ciencia e Innovación de España.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 439-452, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

En paralelo a este crecimiento de Internet, ha habido un incremento muy importante del volumen de información y de datos digitales. Este crecimiento ha llegado hasta tal punto que las organizaciones se han visto con la necesidad de usar herramientas de gestión sólidas para mantener sus aplicaciones Web y gestionar su contenido [3] [2]. Una de las soluciones adoptadas con más éxito ha sido basar el desarrollo de estas aplicaciones Web sobre Sistemas de Gestión de Contenidos (*Content Management System, CMS*) [4]. Estas aplicaciones Web son más fáciles de mantener y permiten a los usuarios recopilar, administrar y publicar contenido de forma integral [5].

En los últimos años, la Ingeniería Web ha proporcionado métodos que han mejorado la eficacia de los procesos de desarrollo Web [6]. La mayoría de estos métodos, tales como WebML [7], OOH [8], UWE [9], OOHDM [10] o HM^3 [11], siguen el paradigma de la Ingeniería Dirigida por Modelos (*Model Driven Engineering, MDE*) para mejorar el desarrollo. La MDE impulsa el uso de modelos como pieza principal en el proceso de desarrollo. De este modo, es posible abordar los detalles de la plataforma de implementación utilizando enfoques basados en la MDE ya que combinan, por un lado, los Lenguajes de modelado Específicos de Dominio (*Domain Specific Language, DSL*) que reflejan la estructura, el comportamiento y los requisitos de un dominio dado; y por otro lado, los mecanismos de transformación que permiten transformar el modelo en diferentes implementaciones [12].

Es difícil encontrar un método Web dirigido por modelos que sea capaz de modelar e implementar correctamente aplicaciones Web de distintos ámbitos. Por esta razón, estos métodos necesitan ser adaptados, ampliados o redefinidos a los diferentes contextos que van surgiendo. Por ejemplo, con la llegada de las RIA (*Rich Internet Applications*) [1], los métodos como WebML, UWE, OOHDM u OOH tuvieron que adaptar sus lenguajes de modelado para que estos tuviesen la expresividad necesaria para modelar el nuevo contexto RIA [2].

Ahora surge el reto de adaptar los métodos Web dirigidos por modelos a los aspectos de los CMSs. Algunos de estos aspectos son: la separación estricta entre contenido, estructura y diseño gráfico; la utilización de un repositorio para la reutilización de la información; la definición de flujos de trabajo para el proceso de creación y publicación de información; la extensibilidad y modularidad; el manejo de diferentes tipos de contenido y la gestión dinámica de la estructura y el aspecto visual [2, 4]. Consideramos que uno de los inconvenientes más importantes que todavía está por resolver es el hecho de que los métodos de desarrollo Web (en concreto, sus lenguajes de modelado), carecen de expresividad para definir aplicaciones Web basadas en CMS [15].

Para contribuir a solventar este problema, en este artículo se presenta un Metamodelo Común para CMS (*CMS Common Metamodel, CMS-CM*) que captura los conceptos necesarios para el modelado de las aplicaciones Web basadas en CMS. Este metamodelo podría ser utilizado como base para adaptar o extender los lenguajes de modelado de los métodos Web existentes, así como para definir nuevos lenguajes de modelado para nuevos métodos específicos en el contexto del desarrollo de aplicaciones Web basadas en CMS.

Para definir el CMS-CM, se han analizado y capturado los elementos clave de tres de las soluciones tecnológicas para CMS de código abierto más utilizadas

actualmente [16]: Drupal [17], Joomla![18] y Wordpress [19]. Estas tecnologías proporcionan los medios para poder implementar aplicaciones Web minimizando el tiempo de desarrollo y mejorando su estabilidad [1].

Para la validación del metamodelo que se presenta, se ha utilizado un caso de estudio basado en una aplicación Web sobre salud y bienestar llamado WebSana. Este caso de estudio nos ha permitido identificar en la Web aquellos elementos recogidos en el CMS-CM.

El resto del articulo se organiza de la siguiente manera: la Sección 2 presenta los trabajos actuales relacionados con el contexto de las aplicaciones Web basadas en CMS; la Sección 3 explica la metodología seguida para definir el CMS-CM; en la Sección 4 se presentan los elementos capturados en el CMS-CM; en la Sección 5 se describe el caso de estudio utilizado para validar el metamodelo; y en la Sección 6 se exponen las conclusiones y los trabajos futuros.

2 Trabajos relacionados

En la última década, han surgido varios métodos para el desarrollo de aplicaciones Web. Muchos de ellos, como es el caso de WebML, OOH, UWE, OOHDM o HM^3, se basan en el paradigma de la MDE.

La mayoría de estos métodos establecen cuatro puntos de vista que recogen los conceptos necesarios para modelar y desarrollar aplicaciones Web [20]. Estas vistas son: la vista de contenido, de navegación, del proceso y de presentación.

Además, éstos definen lenguajes de modelado para un ámbito determinado a través de anotaciones y abstracciones [1]. Los conceptos clave de estos lenguajes se representan en metamodelos. Es posible estructurar estos metamodelos en un conjunto de vistas. Por ejemplo, UWE proporciona un lenguaje de modelado basado en un metamodelo que es una extensión del metamodelo de UML 2.0 que se encuentra estructurado en seis vistas muy similares a las explicadas anteriormente: vista de requisitos, de contenido, de navegación, del proceso, de presentación y de adaptación [21].

Es importante tener presente que el campo de la Ingeniería Web está siempre en evolución y continuamente surgen nuevos escenarios (tecnologías emergentes, arquitecturas Web mejoradas, nuevos paradigmas). Por lo tanto, los métodos de desarrollo Web se ven constantemente con la necesidad de extender sus lenguajes de modelado o de introducir nuevos modelos en su proceso de desarrollo para ser capaces de adaptarse a esta realidad cambiante.

Uno de los escenarios que ha ganado relevancia en los últimos años ha sido el de los CMS. Estos sistemas se han convertido en esenciales para las organizaciones que tienen una significativa presencia en Internet y que manejan grandes volúmenes de contenido digital [3]. Por lo que, muchas de estas empresas han implementado sus aplicaciones Web sobre CMS. Estas aplicaciones Web basadas en CMS presentan un conjunto de características específicas que difieren de las aplicaciones Web tradicionales. Algunas de estas características, de acuerdo con [5] y [3], son las siguientes:

- El elevado número de usuarios productores de información. La gestión del contenido ya no recae únicamente sobre el *webmaster* de la aplicación sino que ahora recae sobre otros roles de la empresa como son clientes, proveedores, socios y personal.
- La identificación y administración de los roles de usuario y la identificación de grupos de usuarios.
- La gestión de diferentes tipos de contenido. Los posibles tipos de contenido que puede gestionar son: HTML, XML, imágenes, documentos, audio, vídeo, etc.
- El almacenamiento de los datos en repositorios y fuentes de datos.
- La clasificación del contenido mediante metainformación.
- La internacionalización. La capacidad de la aplicación Web por adaptarse a idiomas distintos.

También es importante considerar los aspectos recogidos en [22]: uso de la extensibilidad y modularidad; la independencia entre el contenido y la presentación; la gestión dinámica de la estructura y el aspecto visual de la aplicación Web; y la definición del flujos de trabajo.

A causa de la popularidad de las aplicaciones Web basadas en CMS en los últimos años, la comunidad Web ha manifestado un creciente interés en esta nueva plataforma de implementación. Uno de los hechos más significativos es que ya algunos métodos Web dirigidos por modelos están tratando de adaptarse al modelado e implementación automática de este tipo de aplicaciones Web. Además, han surgido nuevos métodos Web específicos para el desarrollo de estas aplicaciones. A continuación, se describen algunos de los trabajos actuales que se centran en este tema.

En [4] los autores presentan un lenguaje gráfico para el modelado de aplicaciones Web basadas en CMS. Este lenguaje de modelado se estructura en dos niveles de abstracción, el CMS-ML (*CMS-Modeling Language*) y el CMS-IL (*CMS-Intermediate Language*). El CMS-ML hace referencia a la especificación de alto nivel de una aplicación Web tradicional y es lo suficientemente simple como para ser utilizado por un usuario no técnico. Con él se pueden definir dos modelos: el *template model* y el *toolkit model*. El *template model* refleja la estructura y el comportamiento de la aplicación Web mientras que el *toolkit model* permite la adición de nuevos elementos que pueden ser utilizados en el *template model*.

El CMS-IL está definido a un nivel menor de abstracción específico para la plataforma CMS en la que se basará la aplicación Web. Éste es utilizado por el diseñador del sistema.

Este lenguaje de modelado se define mediante dos sintaxis: sintaxis abstracta y sintaxis concreta. La sintaxis abstracta se especifica mediante un metamodelo estructurado en diferentes vistas. Por ejemplo, para la parte del CMS-ML se definen trece vistas: estructura (macro y micro estructura), roles, permisos, usuarios, idioma, contenido, aspecto visual, tareas, dominio, estados, componentes Web, interfaz de tareas y efectos secundarios. Los dos principales inconvenientes de este enfoque son la falta de expresividad para hacer frente a los detalles de implementación y el elevado número de vistas que estructuran la sintaxis abstracta.

En [15], se presenta un trabajo que se centra en la mejora y adaptación del método OOWS en el dominio de CMSs. Se evalúa el método OOWS para detectar sus limitaciones en este ámbito. Siguiendo un proceso de Ingeniería de Método Situacional (*Situational Method Engineering, SME*) [23], se extiende el metamodelo de OOWS para integrar los conceptos del dominio de CMS.

Por último, en [1], se presenta WEM (*Web Engineering Method*) como un enfoque para la configuración automática de aplicaciones Web basadas en CMS. Como parte de este enfoque, también proponen una sintaxis abstracta y concreta para modelar este tipo de aplicaciones Web.

3 Enfoque de Modelado

Para la definición del CMS-CM hemos analizado los elementos principales de tres de las tecnologías CMS de código abierto más utilizadas hoy en día en el mercado [16]: Drupal, Joomla! y Wordpress. Siguiendo los principios de la Ingeniería Web hemos estructurado el metamodelo en los cinco aspectos siguientes.

1. **Aspecto de navegación**: este aspecto abarca tanto los elementos que permiten la navegación en la propia aplicación Web basada en CMS.
2. **Aspecto de presentación**: este aspecto representa los elementos que definen la estructura interna de cada página y su apariencia.
3. **Aspecto de comportamiento**: este aspecto contiene los elementos que permiten definir las diferentes funciones desempeñadas por la aplicación.
4. **Aspecto de contenido**: este aspecto hace referencia a los datos que gestiona la aplicación Web.
5. **Aspecto de usuario**: en este aspecto se definen todos los elementos relacionados con los roles, los usuarios y sus permisos.

El proceso que hemos seguido para definir el CMS-CM se compone de cinco actividades principales: 1) el análisis de la plataforma CMS, 2) la definición de un metamodelo para cada tecnología analizada, 3) la definición de los cinco aspectos, 4) la definición del CMS-CM y finalmente 5) la implementación del metamodelo. A continuación se explican cada una de las actividades en detalle.

1. **Análisis de la plataforma CMS:** esta actividad se centra en analizar las tecnologías CMS de código abierto existentes en el mercado para obtener una mejor comprensión del contexto. Gracias a esta actividad se llega a la conclusión de que en la actualidad, las tecnologías CMS de código abierto más utilizadas son Drupal, Joomla! y Wordpress.
2. **Definición de un metamodelo para cada tecnología CMS analizada:** después de analizar y familiarizarnos con estas tecnologías CMS, definimos un metamodelo para cada una de ellas. La definición de estos metamodelos nos permitieron comprender mejor sus características ya que cada metamodelo refleja los elementos clave, sus atributos y la relación que hay entre ellos de cada una de las tecnologías.
3. **Definición de los cinco aspectos:** después de la definición de los metamodelos para cada tecnología CMS, se especificó el conjunto de

aspectos necesarios para agrupar y clasificar los elementos de los metamodelos.

4. **Definición del CMS-CM:** a continuación se procedió a construir el CMS-CM. Para su definición se escogieron los elementos más relevantes de las tres tecnologías analizadas
5. **Implementación del CMS-CM:** después de definir el CMS-CM, se llevó a cabo su implementación mediante el *framework* de modelado de Eclipse (EMF) [24].

4 Metamodelo Común para CMS (CMS-CM)

En esta sección se presentan los elementos que componen el CMS-CM que se muestra en la Fig. 1. A continuación se explican en detalle las clases que aparecen en el metamodelo siguiendo la clasificación en los cinco aspectos definidos.

4.1 Aspecto de navegación

Este aspecto representa los elementos necesarios para definir la estructura de navegación de la aplicación Web basada en CMS. De este modo, se incluye el concepto de *Page* que hace referencia a cada uno de los nodos navegacionales (páginas) que conforman la aplicación. Esta clase dispone de un conjunto de atributos. Cuenta con el atributo *id* que es un identificador único; el atributo booleano *home* que especifica si la página es el nodo raíz de la estructura de navegación y el atributo *urlAlias* que es el nombre por el cual será llamada la página. Además, dispone de tres atributos que definen la visualización de algunos de los datos del contenido que muestra la página. Estos atributos son: el atributo booleano *showName* que especifica si el nombre del contenido que se muestra es visible o no y los atributos *showDate* y *showAuthor* que deciden si visualizar o no la fecha de creación del contenido y el autor que lo creó, respectivamente.

La navegación entre estas páginas es posible mediante el concepto de *Menu* el cual está compuesto por un conjunto de elementos *MenuItem* que representan los enlaces que permiten acceder a cada una de las páginas. Estos elementos están provistos de atributos. Ambos disponen de un *id*, el atributo *name* y el atributo *description*. Además, el elemento MenuItem cuenta con el atributo *weight*, para definir la posición que ocupará dentro de la secuencia de elementos *MenuItems*; el atributo *externalLink* para definir enlaces a páginas externas a la aplicación Web modelada; el atributo booleano *expanded* para hacer visible la jerarquía de elementos *MenuItem*, en el caso de que existiera; y el atributo *status* que define el estado del elemento. Para definir los valores de este último atributo se ha creado un tipo enumerado llamado *StatusT* que especifica los valores de: *published*, *unpublished* y *draft*.

4.2 Aspecto de presentación

En este aspecto se presentan los elementos que definen la estructura interna de las páginas y el aspecto visual de éstas.

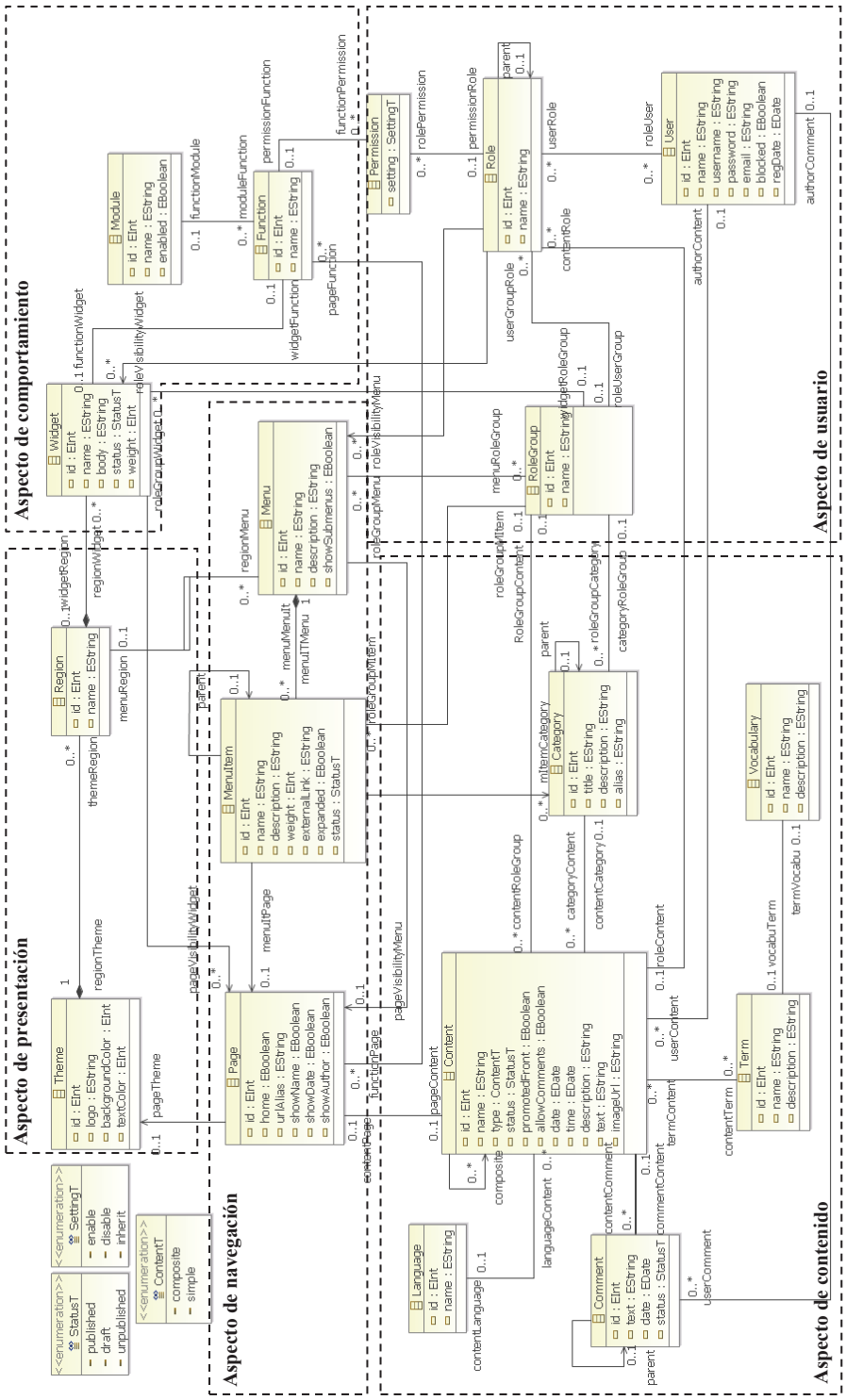


Fig. 1. Metamodelo Común para CMS (*CMS-CM*)

La definición de su apariencia está recogida con el elemento *Theme* que especifica cuál será el logo de la aplicación mediante el atributo logo; el color de fondo de las páginas con el atributo *backgroundColor* o el color del texto con el atributo *textColor*.

Las diferentes áreas que determinan la estructura interna de la página se definen con los elementos *Region*. De esta manera, un *Theme* es una composición de regiones que distribuirán la información mostrada en la página. Éstas pueden albergar elementos *Menu* y elementos *Widget* (estos dos elementos se van a explicar en las siguientes secciones).

4.3 Aspecto de comportamiento

En este aspecto se presentan los elementos que definen la funcionalidad de la aplicación Web. Una de las características principales de las aplicaciones Web basadas en CMS es la modularidad de sus funcionalidades. Esto hace que estas aplicaciones sean más extensibles, escalables; y que se puedan adaptar mejor a las necesidades de los usuarios.

Como se puede observar en Fig. 1, las funcionalidades ofrecidas por la aplicación Web se representan con el elemento *Function*. Éstas se encuentran agrupadas en módulos representados por el elemento *Module*.

Estas aplicaciones ofrecen unos componentes que pueden ser colocados en las diferentes regiones que estructuran la página. Estos componentes aparecen en el CMS-CM con el nombre de *Widgets* y ofrecen una cierta funcionalidad a los usuarios. Disponen de un conjunto de atributos que permiten su configuración. De este modo, el atributo *weight* determina su posición dentro de la región donde se encuentra albergado, y su estado se define mediante el atributo *status* del tipo enumerado *StatusT*. El desarrollador puede implementar en código HTML sus propios *Widgets*. Para ello existe el atributo *body* que almacena este código.

Un aspecto importante de este elemento es la configuración de su visibilidad. Ésta se describe mediante las relaciones existentes con otras clases del metamodelo. Por ejemplo, existe una relación entre la clase *Widget* y *Page* para especificar en qué páginas será visible el componente. Además, existe una relación entre la clase *Widget* y *Role* para definir qué roles de usuario tendrán la posibilidad de visualizarlo. Del mismo modo, ocurre con la relación entre *Widget* y *RoleGroup*.

4.4 Aspecto de contenido

Este aspecto representa uno de los conceptos principales de un *CMS*, el contenido. Éste se recoge en el metamodelo con el elemento *Content,* provisto de una serie de atributos. Algunos atributos como *id*, *name*, *description*, *date* y *time* sirven para definirlo.

Además, dispone del atributo *type* del tipo enumerado *ContentT* que admite los valores *simple* y *composite*. Un contenido del tipo *composite* está formado por otros del tipo *simple*. Por ejemplo, un artículo es un contenido del tipo *composite* ya que está constituido por texto e imágenes que son del tipo *simple*. Además, está provisto de un atributo llamado *status* del tipo *StatusT*.

Tabla 1. Correspondencia de elementos del CMS-CM

	$MM_{CMS\ Common}$	MM_{Drupal}	$MM_{Joomla!}$	$MM_{Wordpress}$
Aspecto de navegación	Page	✓	✓	✓
	Menu	✓	✓	✓
	MenuITem	✓	✓	✓
Aspecto de presentación	Theme	✓	✓ (*Template*)	✓
	Region	✓	✓	✓
Aspecto de comportamiento	Function	✓	✓	✓ (*Task*)
	Widget	✓ (*Block*)	✓ (*Module*)	✓
	Module	✓	✓ (*Component*)	✓ (*Plugin*)
Aspecto del contenido	Content	✓	✓	✓
	Term	✓		✓ (*Tag*)
	Vocabulary	✓		
	Comment	✓		✓
	Language		✓	
	Category		✓	✓
Aspecto del usuario	Role	✓	✓	✓
	RoleGroup		✓ (*ViewingAccesLevel*)	
	Permission	✓	✓	✓ (*Capability*)
	User	✓	✓	✓

Por otra parte, existe un atributo booleano llamado *promotedFront*. Si este atributo se define como verdadero el contendido será visualizado en la página designada como *home*.

Además, el atributo *allowComments* permite al usuario que visualiza el contenido añadir comentarios acerca de éste. Los comentarios están representados en el metamodelo con el elemento *Comment*.

El contenido en un CMS puede ser clasificado. Para ello, existe el elemento *Category* que está asociado al elemento *Content*. Además, es posible añadir metainformación al contenido mediante la inserción de etiquetas. Éstas se representan con el elemento *Term* y pueden ser agrupadas mediante el elemento *Vocabulary*. El contenido puede estar asociado a un idioma en concreto. Por este motivo, en la Fig. 1 aparece la clase *Language* relacionada con el elemento *Content*.

4.5 Aspecto de usuario

Este aspecto abarca los elementos que representan los roles existentes en la aplicación Web y los permisos que a éstos se les asigna. Como se puede ver en la Fig. 1, para cada elemento *Function* se definirá un permiso dado un role. El concepto de role se representa con el elemento *Role* y se identifica con un *id* y el atributo *name*. Por otra parte, el permiso se recoge con el elemento *Permission* que dispone de un atributo llamado *setting* que puede adoptar los valores especificados en el tipo enumerado *SettingT* (*enable*, *disable* e *inherit*).

Los usuarios que pertenecen a un cierto role en la aplicación se representan con el elemento *User*. Para este elemento se definen los atributos *id*, *name*, *username*, *email* y *password*; además de un atributo que almacena la fecha de registro, *regDate* y el estado del usuario con el atributo booleano *blocked*.

Los roles pueden ser agrupados. Para ello, existe el elemento *RoleGroup* que define estos grupos de roles.

La Tabla 1 presenta los elementos recogidos en el CMS-CM y la correspondencia de éstos con cada una de las tecnologías CMS analizadas. Entre paréntesis aparece el nombre específico que adopta el elemento para una tecnología.

5 Caso de estudio

En esta sección se presenta el caso de estudio utilizado para validar los elementos (con sus atributos y relaciones) que aparecen en el CMS-CM. Se trata de demostrar que los elementos capturados en este metamodelo aparecen en una página Web implementada en Drupal (una de las tecnologías analizadas para este trabajo). La aplicación Web que se ha implementado (llamada *WebSana*) sirve para proporcionar información sobre bienestar, deporte, hábitos de nutrición y otros temas relacionados con la salud.

A esta aplicación Web pueden acceder dos tipos de usuarios: usuarios anónimos y socios. A cada grupo se le asigna un conjunto de permisos. A continuación se explican las acciones que puede realizar cada uno de los roles.

Los usuarios anónimos tienen la posibilidad de registrarse como socios y, además, pueden consultar las siguientes páginas informativas: página *Diet* (Dieta) que proporciona una guía de dieta saludable basada en frutas y verduras; página *Sport* (Deporte) que sugiere un conjunto ejercicios físicos recomendados y una explicación sobre la importancia de hacer deporte para mantenerse en forma y página de *Health* (Salud) que contiene consejos sobre cómo prevenir el estrés y la ansiedad en nuestra vida diaria.

Por otro lado, los socios tienen acceso a una encuesta *Is your diet balanced?,* (¿Su dieta es equilibrada?), pueden conectarse a su cuenta de *Twitter* desde *Websana*, pueden solicitar una nueva contraseña, consultar qué usuarios están conectados y qué usuario son nuevos en la aplicación Web. Además, tienen la capacidad de consultar las páginas que ofrecen información comercial y que se explican a continuación: página *Foot and food* (andar y alimentos) que promociona un kit, compuesto por un equipamiento para hacer deporte y un libro para aprender a seguir una dieta sana y equilibrada y página *Vegetarian diet* (dieta vegetariana) que anuncia un paquete de veinte lecciones que explican paso a paso cómo adquirir una dieta vegetariana de forma segura.

Ambos roles son capaces de buscar información dentro de la aplicación Web y comprobar el tiempo gracias a un reloj digital.

Después de definir el caso de estudio, se ha implementado utilizando una de las tecnologías CMS analizadas: Drupal. Por razones de espacio, sólo presentamos algunos de los conceptos capturados en el CMS-CM y su representación en la implementación en Drupal.

5.1 Menu y MenuItem

Como se ha visto anteriormente el elemento *Menu* permite la navegación a través de la aplicación Web. Este elemento se compone de diferentes *MenuItems*, siendo estos los enlaces a las distintas páginas. La Fig. 2 muestra la representación de los elementos *Menu* y *MenuItem* sobre una implementación en Drupal. Se observa un menú compuesto por los elementos *MenuItem* siguientes: *home*, is *your diet balanced?*, *diet*, *sport and health*.

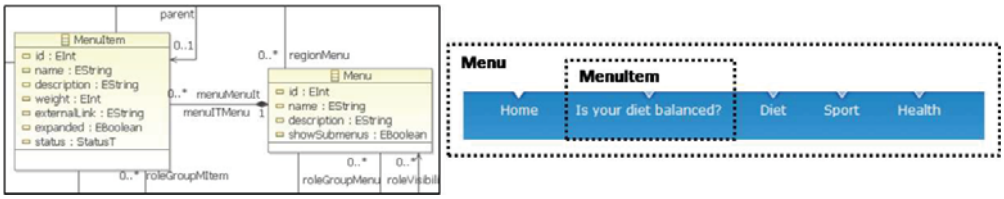


Fig. 2. Elementos Menu y MenuItem

5.2 Content y Term

Como se ha explicado en el apartado anterior el elemento *Content* representa aquella información que será visualizada en una página de nuestra aplicación Web. A este contenido se le puede asociar metainformación mediante un conjunto de etiquetas representadas con el elemento *Term*. La Fig. 3 muestra los elementos *Content* y el *Term* sobre la implementación en Drupal. En la figura se visualiza un artículo (un tipo de *Content*), compuesto de texto, una imagen y un conjunto de *Terms* (*fitness*, salud, deporte y ejercicios).

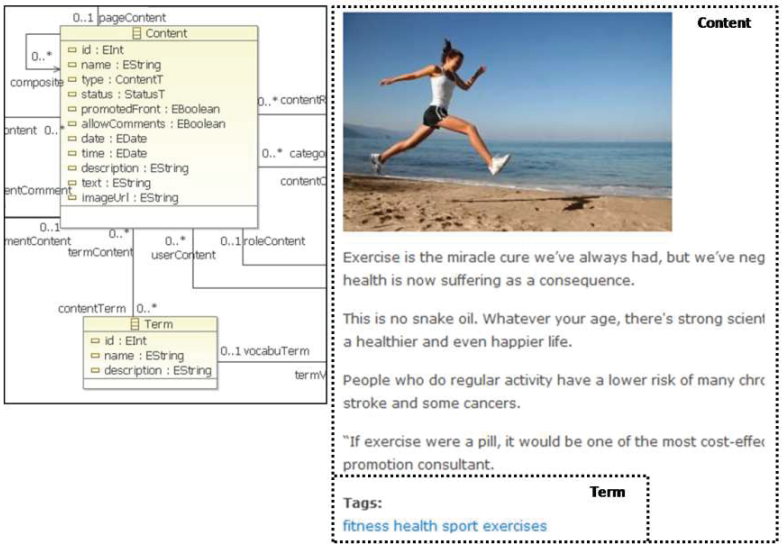


Fig. 3. Elementos Content y Term

5.3 Theme, Region y Widget

Como se ha dicho anteriormente, el elemento *Theme* recoge la información necesaria para definir la apariencia de la página Web, además de definir la estructura general de la página Web en regiones (*Regions*). Por otro lado, las regiones pueden contener diferentes elementos *Widget*.

La Fig. 4 muestra la implementación en Drupal de los elementos *Theme*, *Region* y *Widget*. En la figura aparecen marcados en negrita las regiones (*Upper Region*, *Left Region*, *Center Region* y *Right Region*), el concepto de *Theme* (apariencia de la página Web + conjunto de regiones) y un elemento *Widget* en forma de reloj situado en la región derecha (*Right Region*) que permite a los usuarios consultar la hora.

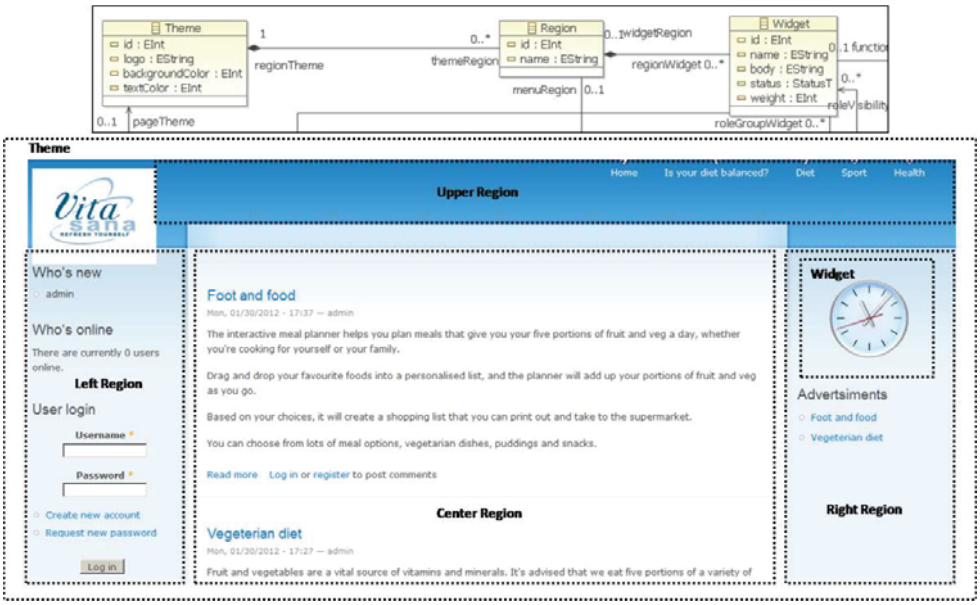


Fig. 4. Elementos Theme, Region y Widget

5 Conclusiones y Trabajos Futuros

Hoy en día muchas empresas apuestan por las aplicaciones Web basadas en CMS, ya que les permite gestionar con eficacia el contenido que manejan, y los usuarios pueden acceder fácilmente a esa información a través de una intranet, extranet o Internet [3].

A causa de la creciente popularidad de las aplicaciones Web basadas en CMS la comunidad Web ha manifestado su interés en esta nueva plataforma de implementación. En el entorno de la MDE algunos métodos Web dirigidos por modelos como OOWS han tratado de adaptar su proceso al modelado e implementación automática de este tipo de aplicaciones Web.

Además, han surgido nuevas propuestas como lenguajes de modelado centrados en el dominio CMS (*CMS-ML y CMS-IL)*, y métodos dirigidos por modelos específicos para el desarrollo de estas aplicaciones como *WEM* (*Web Engineering Method*).

A pesar de ello, las propuestas existentes son todavía limitadas e incompletas, sobre todo a lo que se refiere a los lenguajes de modelado. Por este motivo, en este artículo presentamos un metamodelo común para CMS (*CMS Common Metamodel, CMS-CM*) que recoge los conceptos clave de este ámbito y que puede ayudar a enriquecer los lenguajes de modelado existentes y hacerlos capaces de modelar aplicaciones Web basadas en CMS. Además, este metamodelo podría servir de base para nuevos lenguajes específicos en este ámbito.

Uno de los aspectos más interesantes de este metamodelo es que para su definición se han tenido en cuenta tres de las soluciones tecnológicas de código abierto para CMS más utilizadas hoy en día en el mercado: Drupal, Joomla! y Wordpress.

Se han definido cinco aspectos para la clasificación de los conceptos capturados en el CMS-CM: aspecto de navegación, presentación, de comportamiento, de contenido y de usuario. Éstos son muy similares a las vistas propuestas por la Comunidad Web para el desarrollo de aplicaciones Web.

En la actualidad, estamos trabajando en tres líneas: 1) la definición de las transformaciones automáticas que generen a partir de modelos ya conocidos hasta el momento (modelo de negocio, modelo de navegación, modelo conceptual de datos, etc.) un modelo que contenga los conceptos capturados en el CMS-CM que presentamos, 2) la definición de las transformaciones automáticas que transformen el modelo CMS en código, y 3) la integración del CMS-CM en lenguajes de modelado ya existentes.

Referencias

1. Souer, J., Kupers, T., Helms, R., and Brinkkemper, S.: Model-Driven Web Engineering for the Automated Configuration of Web Content Management Systems. In: ICWE 2009, pp. 124-135. Heidelberg (2009)
2. Souer, J., van de Weerd, I., Versendaal, J., and Brinkkemper, S.: Situational Requirements Engineering for the Development of Content Management System-based Web Applications. In: International Journal of WET vol. 3, pp. 420 - 440 (2007)
3. McKeever, S.: Understanding Web content management systems: Evolution, Lifecycle and Market. In: Industrial management + data systems, vol. 103, pp. 686 - 692 (2003)
4. de Sousa, J. and Rodrigues, A.: Web Application Modeling with the CMS-ML Language. In: Inforum, pp. 461-472. (2010)
5. Boiko, B.: Understanding Content Management. In: Bulletin of the American Society for Information Science and Technology, vol. 28, pp. 8-13 (2001)
6. Deshpande, Y., et al.: Web Engineering. In: Journal of WE, vol. 1, pp. 3-17 (2002)
7. Ceri, S., Fraternali, P., and Bongio, A.: Web Modeling Language (WebML): a Modeling Language for Designing Web Sites. In: Computer Networks, vol. 33, pp. 137-157 (2000)
8. Gómez, J., Cachero, C., and Pastor, O.: Extending a conceptual modelling approach to Web application design. In: International Conference on Advanced Information Systems Engineering, pp. 79-93, (2000)

9. Koch, N. and Kraus, A.: The expressive power of UML-based Web Engineering. In: International Workshop on Web Oriented Software Techonology (2002)
10. Schwabe, D. and Rossi, G.: The Object Oriented Hypermedia Design Model. In: Communications of ACM, vol. 38, pp. 45-46 (1995)
11. Cáceres, P., V., D.C., J.M., V., and Marcos, E.: Model transformations for hypertext modeling on web information systems. In: SAC, vol., pp. 1232-1239 (2006)
12. Schmidt, D.: Model-Driven Engineering. In: IEEE Computer, (2006)
13. Preciado, J.C., Linaje, M., Comai, S., and Sanchez-Figueroa, F.: Necessity of Methodologies to Model Rich Internet Applications, in 6th International Symposium on Web Site Evolution. 2005.
14. Fraternali, P., Comai, S., Bozzon, A., and G., T.: Engineering Rich Internet Applications with a Model-Driven Approach. In: ACM Transactions on the web, vol. 4, pp. 47 (2010)
15. Vlaanderen, K., Valverde, F., and Pastor, O.: Model-Driven Web Engineering in the CMS Domain: A Preliminary Research Applying SME. In: Enterprise Information Systems, vol. 19, pp. 226-237, Heidelberg (2009)
16. Shreves, R.: Open Source CMS Market Share, W.S. White paper, Editor. 2011.
17. Drupal CMS, http://drupal.org/
18. Joomla! CMS, http://www.joomla.org/
19. Wordpress CMS, http://wordpress.org/
20. Kraus, A., Knapp, A., and Koch, N.: Model-Driven Generation of Web Applications in UWE. (2008)
21. Kroiss, C. and Koch, N.: The UWE Metamodel and Profile - User Guide and Reference. 2008, Ludwig-Maximilians-Universität München, Institute for Informatics: München.
22. de Sousa, J. and Rodrigues, A.: CMS-based Web-Application Development Using Model-Driven Languages. In: Fourth International Conference on Software Engineering Advances, pp. 500-505. IEEE Computer Society, Porto, Portugal (2009)
23. Ralyté, J., Deneckère, R., and Rolland, C.: Towards a generic model for situational method engineering. In: CAISE 2003, pp. 1029. Springer, Heidelberg (2003)
24. Budinsky, F.: Eclipse Modeling Framework. 2008, Addison-Wesley Professional.
25. Corlosquet, S., et al.: Produce and Consume Linked Data with Drupal! . In: ISWC 2009, vol. 5823/2009, pp. 763-778 (2009)

Moskitt4SPL: Tool Support for Developing Self-Adaptive Systems

María Gómez, Ignacio Mansanet, Joan Fons, and Vicente Pelechano

Centro de Investigación en Métodos de Producción de Software
Universitat Politècnica de València
Camino de Vera s/n, E-46022, Spain
{magomez,imansanet,jjfons,pele}@dsic.upv.es

1 Introduction

Increasingly, software needs to dynamically adapt its behavior at runtime in response to changing conditions in the supporting computing, communication infrastructure, and in the surrounding physical environment [6]. Self-adaptive software systems are able to reconfigure at run-time to adapt to changes. The implementation of ad-hoc solutions to cover all possible system configurations and reconfigurations is not feasible. Dynamic Software Product Lines (DSPLs) provide a systematic basis for the engineering of self-adaptive systems [4]. A key characteristic in DSPLs is the intensive use of variability at run-time in order to adapt the system configuration caused by an environment change. Following this approach, a self-adaptive system can be seen as a family of feasible system configurations with a mechanism to move from one configuration to another. The development of self-adaptive systems involves great complexity and becomes a tedious task. We propose *Moskitt4SPL (M4SPL)* an open-source tool to ease the development of self-adaptive systems. In this tool, we combine model-driven and DSPLs to better cope with the complexities during the construction of self-adaptive systems. M4SPL can be used for modeling systems which make use of variability at run-time in order to adapt the system configuration caused by an environment change. M4SPL provides edition capabilities for Feature Models, Configuration Models and Resolution Models which are part of a self-adaptive system specification. Furthermore, M4SPL incorporates a series of refinements to automatically ensure interesting behavior issues in adaptation specifications. Dealing with those issues before execution is essential for reliable self-adaptive systems that fulfill many of the user's needs. M4SPL can be used standalone as an Eclipse plug-in or integrated in the MDE MOSKitt environment.

2 M4SPL Tool: Overview

Moskitt4SPL (M4SPL) is a free open-source tool which gives support for modeling self-adaptive systems. M4SPL is based on Eclipse plug-ins: Eclipse Modeling Framework (EMF) [1], Graphical Modeling Framework (GMF) [2] and Atlas Transformation Language (ATL) [5]. The major novel feature of this tool is the application of model-driven and product-line engineering for designing self-adaptive systems. M4SPL provides different editors to ease the specification of self-adaptive systems:

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 453-456, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

– **Feature Model Editor.** Feature Models are a widely used notation to capture the variability of a system in terms of features. Feature Models describe, in an intensional way, the possible configurations in which the system can evolve. M4SPL includes a graphical Feature Model Editor. The Feature Model Editor is based on a previous tool called Mokitt Feature Modeler [3]. Figure 1 (left) illustrates a screenshot of the Feature Model Editor.
– **Feature Model Configuration Editor.** From the defined Feature Model, designers can create Configuration Models. A Configuration Model represents a feasible configuration of the Feature Model. A configuration is defined by the set of states of each feature in the Feature Model. The possible feature states are: active, inactive or discarded. A graphical editor is provided to describe possible configurations of a Feature Model. Figure 1 (right) illustrates the environment of the editor. Features are represented with different colors depending on their state: green features represent active features, red features represent inactive features and orange features represent discarded features.

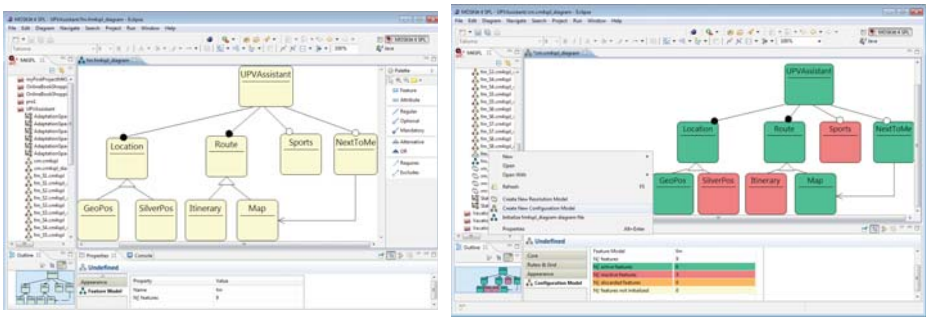


Fig. 1. Feature Model Editor (left). Configuration Model Editor (right)

– **Resolution Model Editor.** Resolutions define reconfigurations among the different system configurations in a declarative manner. Each Resolution is associated with a context condition and represents the sequence of actions in terms of feature activation/deactivation produced in the system when a context change occurs and the condition is fulfilled. An editor is provided to support the definition of the Resolution Model using the EMF capabilities. Figure 2 (left) shows the environment of the editor.

One of the main reasons for following a MDE development is that it is focused on automation. Models can be transformed automatically into new models or code by means of model transformation techniques. This enables automation in system development. In order to automate the design method, we have implemented a series of model-to-model transformations using ATL:

– **Adaptation Space Generation.** Using the tools presented above, designers can define the initial self-adaptive system specification by means of: a Feature Model,

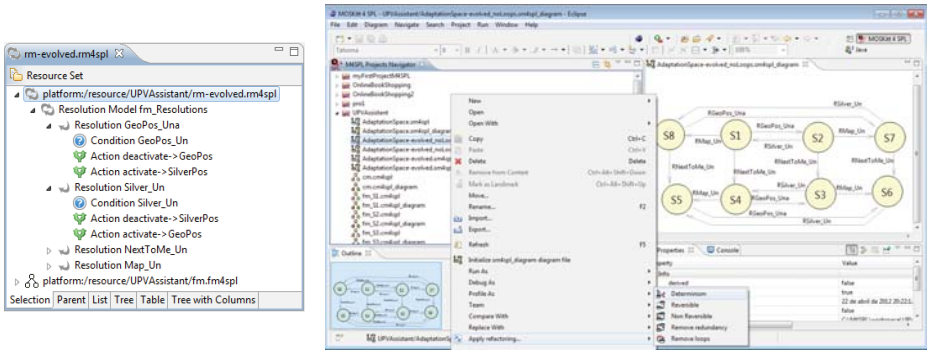


Fig. 2. Resolution Model Editor (left). State Machine Model Editor (right)

an initial Configuration Model and a Resolution Model. This system specification can be automatically transformed into a State Machine Model that represents the implicit Adaptation Space of the system. The state machine representation eases the analysis of the system behavior. The Adaptation Space contains all the feasible configurations that the system can reach through execution, and the reconfigurations among them. The configurations are graphically represented as states and the reconfigurations are graphically represented as transitions. Each state has associated a Configuration Model, and each transition has associated a Resolution of the Resolution Model. Figure 2 (right) shows a screenshot of the Adaptation Space generated by M4SPL from a specification. Once the State Machine Model has been obtained, several refinements can be applied successively to guarantee properties about the adaptive behavior at run-time. Such refinements are made by means of rules implemented in the ATL. All the transformations take as input two models: the State Machine Model and the Resolution Model; and generate a Refined State Machine Model and a Refined Resolution Model.

– **Refinements.** M4SPL enables to analyze the Adaptation Space and to automatically refine the model specifications to ensure behavioral issues. M4SPL supports five refinements:

  - *Determinism:* M4SPL implements a refinement to ensure that the system is determinist. The refinement guarantees that from a given state, when a determined context condition is fulfilled, the system can only reconfigure to one destination state. This refinement modifies the Resolution Model in order to avoid simultaneous reconfigurations in the system.
  - *Reversibility:* M4SPL implements two model refinements to achieve a specification (1) with rollback capabilities or (2) without rollback capabilities. The purpose of the first refinement is to ensure that, for all configurations contained in the Adaptation Space, exist a reconfiguration that leads directly to the previous configuration. The refinement generates new resolutions which define compensation actions to reverse all the reconfigurations. The purpose of the second refinement is to ensure that there is no reconfiguration that leads di-

rectly to the previous configuration. This refinement modifies the Resolution Model in order to avoid reversible reconfigurations

- *Remove Redundancy:* M4SPL provides a refinement to guarantees that the specification does not contain duplicated behavior. This refinement eliminates redundant reconfigurations in a self-adaptive specification. Two (or more) reconfigurations are redundant if they evolve the system from the same source configuration to the same target configuration. The refinement, first finds redundant resolutions. Then, it selects the simplest resolution and removes the others. We consider the simplest resolution as the one that involves the minimum number of change actions. The elimination of redundancy can actually improve execution time and understandability of the system behavior.
- *Remove Loops:* M4SPL implements a model refinement to remove loops in the system specification. The refinement modifies the Resolution Model in order to avoid reconfigurations that evolve from a source configuration to the same configuration.

M4SPL also provides some metrics about the specification (i.e number of active features, number of configurations, number of reconfigurations etc). M4SPL is available online at: `http://www.pros.upv.es/m4spl/`.

Acknowledgments. This work has been supported by Conselleria d'Educación of Generalitat Valenciana under the program VALi+d and by Ministerio de Ciencia e Innovación (MICINN) under the project EVERYWARE TIN2010-18011.

References

1. EMF. Eclipse Modeling Framework, `www.eclipse.org/modeling/emf`
2. GMF. Graphical Modeling Framework, `www.eclipse.org/modeling/gmf`
3. MOSKitt Feature Modeler, `http://www.moskitt.org/eng/proyectomfmmoskittfeaturemod/`
4. Hallsteinsen, S., Hinchey, M., Park, S., Schmid, K.: Dynamic Software Product Lines. Computer 41(4), 93 –95 (2008)
5. Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I.: ATL: A model transformation tool. Sci. Comput. Program. 72(1-2), 31–39 (Jun 2008)
6. McKinley, P.K., Sadjadi, S.M., Kasten, E.P., Cheng, B.H.: Composing adaptive software. Computer 37, 56–64 (2004)

Aplicando los principios del DSDM al desarrollo de transformaciones de modelos en ETL

Álvaro Jiménez, Verónica A. Bollati, Juan M. Vara, Esperanza Marcos

Grupo de Investigación Kybele,
Universidad Rey Juan Carlos, Madrid (España).
{alvaro.jimenez,veronica.bollati
juanmanuel.vara,esperanza.marcos}@urjc.es

Resumen Las transformaciones de modelos son uno de los principales artefactos en el Desarrollo de Software Dirigido por Modelos. Sin embargo, a pesar de ser otro artefacto software más, existen pocas aproximaciones que apliquen los principios del DSDM a su desarrollo. En este trabajo presentamos una aproximación para el desarrollo de transformaciones de modelos dirigido por modelos para el lenguaje *Epsilon Transformation Language* (ETL). Para ello, presentamos un metamodelo para el lenguaje ETL, una transformación que permite obtener un modelo ETL a partir de un modelo de la transformación de alto nivel y la generación del código ETL que implementa la transformación.

Palabras Clave: Desarrollo Dirigido por Modelos, Transformaciones de Modelos, Modelos de Transformación, ETL, ATL.

1. Introducción

En el contexto del Desarrollo de Software Dirigido por Modelos (DSDM, [27], [30]), las transformaciones de modelos actúan principalmente como elementos de enlace entre los diferentes pasos del proceso de desarrollo, refinando los modelos de alto nivel en modelos de menor abstracción, hasta que estos puedan ser ejecutados/interpretados o convertidos en código ejecutable. Además, las transformaciones de modelos pueden emplearse para llevar a cabo otras tareas como la sincronización o la migración de modelos [24],[31]. Por ello, independientemente del objetivo con el que hayan sido construidas, las transformaciones de modelos juegan un papel clave en cualquier propuesta relacionada con la Ingeniería Dirigida por Modelos (*Model-Driven Engineering*, MDE) [26].

Como prueba de ello, durante los últimos años ha surgido un gran número de lenguajes y herramientas que dan soporte al desarrollo de transformaciones [6], [9], [29]. Estos lenguajes y herramientas difieren en múltiples aspectos, como la aproximación que adoptan (declarativa, imperativa, híbrida, basada en grafos, etc.) y esta diversidad trae consigo una complejidad adicional en el desarrollo de transformaciones como la de la selección del lenguaje, el tiempo de aprendizaje, la migración, etc.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 457-470, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

En este sentido, dado que el desarrollo de transformaciones es una actividad intrínsecamente compleja [13] y que cada vez existe mayor número de alternativas para la construcción de las mismas, resultaría recomendable aplicar los mismos principios del DSDM a su desarrollo [3], [11], [19]. De esta forma se podrían modelar las transformaciones a alto nivel y refinar dichas especificaciones produciendo modelos de bajo nivel. Además, al ser capaces de expresar las transformaciones en forma de modelos, podremos procesarlas como haríamos con cualquier otro modelo. Por ejemplo, estaremos en condiciones de generar, validar, comparar o refactorizar transformaciones [2]. El modelado de las transformaciones también permitiría establecer puentes tecnológicos entre los diferentes lenguajes de transformación, ya que un modelo de transformación para un lenguaje concreto se podría transformar en un modelo para otro lenguaje diferente.

En este sentido, existen distintas propuestas que se centran en el modelado de las transformaciones de modelos ([3], [11], [19], [20]). Sin embargo, la mayoría de ellas son sólo propuestas teóricas y no ofrecen soporte tecnológico para llevar a la práctica el modelado de las transformaciones, o al menos no soportan el proceso completo.

Por todo ello, este trabajo muestra la aplicación de la propuesta presentada en trabajos anteriores para soportar el desarrollo dirigido por modelos de transformaciones de modelos ([6], [12]) al desarrollo de transformaciones para el lenguaje ETL (*Epsilon Transformation Language*, [17]).

Así, las principales contribuciones de este trabajo son: la implementación de un metamodelo para el lenguaje de transformaciones ETL; el desarrollo de una transformación que permite mapear modelos de transformación de alto nivel a modelos ETL y el desarrollo de una transformación modelo a texto que, a partir del modelo de una transformación ETL, permite obtener el código que implementa la transformación modelada.

El resto del documento se estructura de la siguiente forma: en la Sección 2 se realiza una breve introducción al lenguaje ETL. La Sección 3 presenta la propuesta para el desarrollo dirigido por modelos de transformaciones ETL; la Sección 4 usa un caso de estudio para ilustrar el resultado y finalmente en la Sección 5 se resumen las principales conclusiones, identificando futuras líneas de investigación.

2. Conceptos previos: Lenguaje de Transformación ETL

ETL (*Epsilon Transformation Language*, [17]) es un lenguaje para el desarrollo de transformaciones de modelos que sigue una aproximación híbrida (estilos declarativo e imperativo). Forma parte de la familia de lenguajes de Epsilon y por lo tanto del proyecto Eclipse GMT (*Generative Modeling Technologies*). Actualmente es usado ampliamente por la comunidad de desarrolladores, proporcionando amplia documentación, ejemplos de uso y un foro de discusión.

El lenguaje ETL potencia el uso de las construcciones declarativas mediante la definición y ejecución de reglas. Sin embargo, para soportar construcciones imperativas, como operaciones o iniciación de variables, se apoya en el lengua-

je EOL (*Epsilon Object Language*, [16]). Para definir la transformación, ETL proporciona distintos tipos de reglas:

- *Matched Rule*: son las reglas principales de la transformación. Permiten definir qué elemento o elementos se deben generar en el modelo destino a partir de la existencia de uno o varios elementos en el modelo origen.
- *Lazy Rule*: son reglas que deben ser invocadas explícitamente por otras reglas.
- *Abstract Rule*: reglas abstractas que pueden ser extendidas por otras reglas.
- *Greedy Rule*: a diferencia de otras reglas, los elementos que las invocan no tienen porqué ser exactamente del mismo tipo definido, sino que también pueden ser de un subtipo de dicho tipo (*Type-of vs. Kind-of*). Por ejemplo, si se tiene un elemento X del que heredan los elementos Y y Z, las reglas *greedy* definidas para X también se ejecutarán sobre los elementos Y y Z.
- *Primary Rule*: para devolver los elementos generados a partir de un elemento concreto, una de las operaciones que proporciona ETL es `equivalents()`. Por defecto, la operación `equivalents()` devuelve una lista de elementos, ordenada según el orden en el que se han definido las reglas que los han creado. Sin embargo, cuando una regla es definida como *primary* sus resultados preceden, en dichas listas, a los del resto de tipos de reglas.

3. Desarrollo Dirigido por Modelos de Transformaciones en ETL

Este trabajo ha sido abordado en el contexto de MeTAGeM-Trace, un entorno para el desarrollo dirigido por modelos de transformaciones de modelos que incluyen generadores de trazas [13]. En concreto, el trabajo aquí presentado constituye parte de la prueba de concepto de MeTAGeM-Trace. En la Fig. 1 se muestra una visión global del proceso propuesto para el desarrollo dirigido por modelos de transformaciones ETL a diferentes niveles de abstracción.

Con el objetivo de aumentar el nivel de abstracción al que se especifican las transformaciones ETL, definimos las transformaciones de acuerdo a los siguientes niveles: PSM, PDM y Código. A nivel específico de plataforma (PSM, *Platform Specific Model*), se modelarán las transformaciones de modelos siguiendo una aproximación de transformaciones de modelos concreta (declarativa, imperativa, híbrida, etc.). En este trabajo, se define un modelo de transformación conforme a un metamodelo que incluye las abstracciones propias de la mayoría de los lenguajes de transformación que siguen una aproximación híbrida como ETL, ATL [14] o RubyTL [25]. Es importante mencionar, que de la misma manera, a este nivel podrían definirse metamodelos para otras aproximaciones, como por ejemplo: declarativo, imperativo, orientado a grafos, etc. A nivel dependiente de plataforma (PDM, *Platform Dependent Model*), se modelaran las transformaciones de acuerdo a un lenguaje de transformación de modelos en concreto, dicho lenguaje, deberá seguir la aproximación seleccionada previamente. En este trabajo, se incluyen los modelos de transformación conformes al metamodelo definido para ETL (Sección 3.1). Por último, a nivel código se obtiene el código que

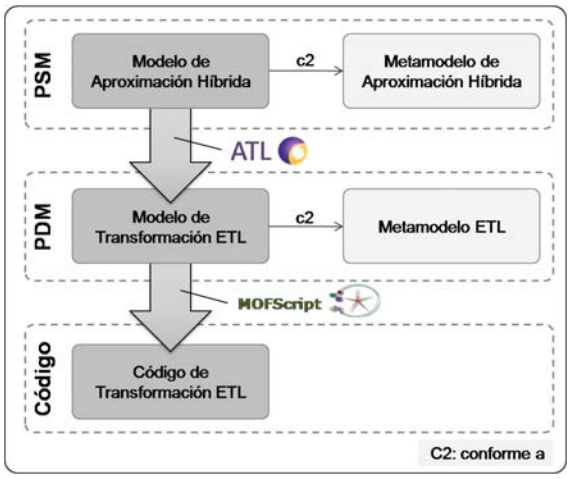


Figura 1. Proceso de Desarrollo Dirigido por Modelos de Transformaciones ETL

implementa la transformación en el lenguaje seleccionado en el nivel anterior, por tanto se obtiene el código ETL que implementa la transformación.

El paso de los modelos entre los niveles de abstracción se realiza mediante la ejecución de transformaciones, en el mapeo entre los modelos PSM y PDM se emplea una transformación de modelo a modelo (M2M), implementada con el lenguaje ATL [14] Para la generación del código ETL (PDM-código) se emplea una transformación de modelo a texto (M2T), implementada con el lenguaje MOFScript [22].

3.1. Metamodelo para ETL

Aunque en la literatura se han encontrado algunas especificaciones de la sintaxis abstracta del lenguaje ETL ([17], [18]), no se ha encontrado ninguna implementación de su metamodelo. Por tanto, para dar soporte al modelado de transformaciones ETL ha sido necesario especificar e implementar un metamodelo para este lenguaje. Para llevar a cabo esta tarea, se ha analizado en profundidad la sintaxis del lenguaje así como ciertos aspectos del lenguaje EOL y los ejemplos disponibles en el sitio web de la herramienta Epsilon. En la Fig. 2 se muestra el metamodelo para el lenguaje de transformaciones ETL implementado en términos de un modelo Ecore [7].

En un modelo Ecore, al utilizar la representación XML, se debe definir un elemento raíz a partir del cual se definan el resto de elementos. En este caso, el elemento raíz del metamodelo de ETL es la metaclase `EtlModule` que representa a la transformación en sí misma. A partir de dicho elemento se pueden definir: reglas de transformación (`TransformationRule`); bloques de código EOL (`EolBlock`) para definir pre-condiciones y post-condiciones de la transformación; y operaciones EOL (`Operation`) que se pueden invocar a lo largo de la transformación.

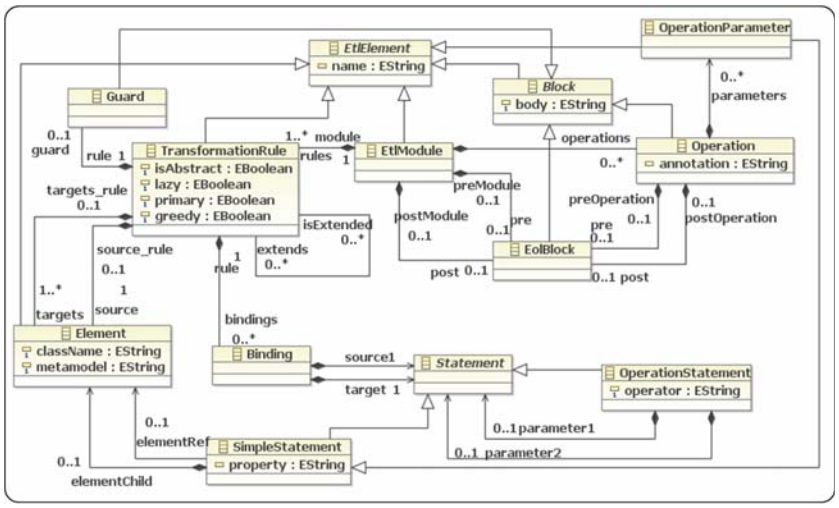


Figura 2. Metamodelo para el lenguaje ETL

Respecto a la definición de las reglas de transformación conviene mencionar que, como se ha descrito en la sección 2, ETL permite la definición de varios tipos de reglas, sin embargo, en el metamodelo definido todas ellas se representan mediante la metaclase `TransformationRule`, en la que se definen atributos booleanos (`isAbstract`, `lazy`, `primary` y `greedy`), que distinguen cada uno de los tipos de reglas. Además, en cada regla se podrán definir los elementos que intervienen en las mismas (`Element`) y asignaciones (`Binding`).

3.2. Transformación de Modelos de Aproximación Híbrida a Modelos ETL

En esta sección nos centramos en la generación de modelos de transformación ETL (conformes al metamodelo presentado en la sección anterior) a partir de modelos de la transformación definidos de acuerdo a las características de la aproximación híbrida. Para ello, se desarrolla una transformación de modelos, denominada `Hybrid2ETL`, que establece las reglas de mapeo entre los elementos del metamodelo Híbrido, definido en [13], y del metamodelo ETL.

En cuanto al desarrollo de esta transformación, aunque en ocasiones se recomienda seguir un proceso de desarrollo completo (elicitación de requisitos, análisis, diseño, implementación y pruebas) [11], en [23] se indica que el mapeo que describen las transformaciones puede realizarse en lenguaje natural, algoritmos o en modelos de mapeo. A partir de la experiencia en trabajos anteriores ([6],[29]) y dado que la definición de las reglas de transformación en lenguaje natural puede considerarse, en cierto modo, como una aproximación a las fases de análisis y diseño, se ha optado por seguir los siguientes pasos para el desarrollo de las transformaciones de modelos:

1. Definir las transformaciones entre los modelos en lenguaje natural.

2. Estructurarlas en un conjunto de reglas de mapeo, expresadas nuevamente en lenguaje natural.
3. Implementar dichas reglas usando un lenguaje de transformación existente. En particular, se ha optado por emplear ATL [15], que es considerado el estándar *de-facto* para el desarrollo de transformaciones de modelos([4], [28]). Además, comparado con otros lenguajes ([6], [29]), en nuestra opinión es el más adecuado debido a la cantidad de documentación disponible, al soporte tecnológico que ofrece y al número de casos de éxito.

De acuerdo a este proceso, en la Tabla 1 se describe, de forma resumida, las reglas de mapeo para transformar modelos de transformación de alto nivel en modelos de transformación ETL.

Metamodelo de Aproximación Híbrida	**Metamodelo ETL**
`Module`	`EtlModule`
`Rule` (`sources`==1 and `targets` > 0)	`Transformation Rule` - `source`: `rule.sources` - `target`: `rule.targets`
`Binding`	`Binding`
`LeftPattern`	`SimpleStatement`
`TransformationElement` (incluido en un `OpDefinition` o en un `OpArgument`)	`Element`
`Source` (incluido en una `Rule`)	`Element` `parent` = `Rule`
`Source` (incluido en un `RightPattern`)	`Element` `parent` = `Element`
`Target` (incluido en una `Rule`)	`Element` `parent` = `Rule`
`Target` (incluido en un `LeftPattern`)	`Element` `parent` = `Element`
`Guard`	`Guard`
`Operation`	`Operation`
`OpDefinition` (`return`==`Boolean` or `return`==`Integer` or `return`==`String`)	`SimpleStatement`
`OpArgument` (`return`==`Boolean` or `return`==`Integer` or `return`==`String`)	`OperationParameter`

Tabla 1. Reglas de mapeo: de modelos de transformación de aproximación híbrida a modelos ETL

Conviene mencionar que debido a su complejidad, esta tabla no incluye el mapeado del elemento `RightPattern`, que puede ser convertido en elementos `SimpleStatement` o `OperationStatement`, dependiendo de sus referencias y del

tipo de relación a la que pertenezca (para más información acerca de estas relaciones, consulte [13]).

El siguiente paso consiste en implementar dichas reglas usando ATL. A modo de ejemplo, a continuación se muestra el código de la regla ATL que implementa el mapeo entre el elemento `Module` del metamodelo de aproximación híbrida y el elemento `EtlModule` del metamodelo de ETL, que se corresponden con los elementos raíz de cada uno de estos metamodelos. Al mismo tiempo que se genera el elemento destino, se generan sus atributos y referencias que representan el nombre, las reglas y las operaciones de la transformación que se está generando (`name`, `rules` y `operations`). En esta implementación, estos elementos se construyen directamente a partir de los elementos de mismo nombre registrados en el modelo de la transformación a alto nivel.

```
rule Module{
from
   h_module: Hybrid!Module
to
   etl_module: ETL!EtlModule(
     name <- h_module.name,
     rules <- h_module.rules,
     operations <- h_module.operations
   ) }
```

Una vez implementada la regla anterior, se procede a implementar el resto de reglas de transformación mostradas en la Tabla 1. Así, por ejemplo, para pasar de elementos de tipo `rule` (alto nivel) a elementos de tipo `TransformationRule` en ETL se ha implementado la regla `createRule`:

```
rule createRule{
from
   r: Hybrid!Rule(r.sources.size()=1 and r.targets.size()>0)
to
   etl_r: ETL!TransformationRule(
     name <- r.name,
     isAbstract <- r.isAbstract,
     "extends" <- r."extends",
     "lazy" <- not r.isMain,
     guard <- r.guard,
     source <- r.sources.first(),
     targets <- r.targets,
     bindings <- r.targets->collect(t|t.bindings).asSequence()
   ) }
```

Dado que ETL no permite definir reglas con más de un elemento de entrada ni reglas sin elementos destino, en la regla de transformación `createRule` se ha definido una condición (o guarda) que impide transformar reglas que no cumplan estas características. Asimismo, conviene destacar la creación de los atributos

`isAbstract`, `extends` y `lazy` que establecen las características de la regla ETL generada.

Como muestra la Fig. 1, ejecutando estas reglas de transformación en el motor de ATL se consume un modelo de la transformación siguiendo la aproximación híbrida y se produce un modelo de la transformación conforme al metamodelo de ETL.

3.3. Generación de Código

El siguiente paso consiste en generar el código que implementa la transformación ETL. Para ello, se construye una transformación M2T, siguiendo los mismos pasos que para la construcción de la transformación M2M anterior: definición en lenguaje natural, estructuración en reglas de mapeo e implementación con un lenguaje de transformaciones.

Metamodelo ETL	**Código ETL**

`EtlModule`	'pre' pre.name '{'
	  pre.body
	'}'
	module.transformationRules
	module.operations
	'post' post.name ' {'
	  post.body
	'}'

`TransformationRule`	[('@greedy ', '@abstract ', '@lazy ', '@primary ')]
	'rule' name
	'transform' source
	'to' targets
	['extends' extends]
	'{'
	  ['guard:' guard.body]
	  bindings
	'}'

`Binding`	target ':=' source ';'
`SimpleStatement(Binding)`	[ref '.'] property
`SimpleStatement(Operation)`	[child.model '!' className] [ref '.'] property
`OperationStatement`	[parameter1] operator [parameter2]
`Element`	name ':' model '!' class
`Operation`	['@' annotation] ['@' pre] ['@' post] 'operation' context ['(' parameters ')'] ': return' return '{' body '}'
`OperationParameter`	name ':' [ref.model '!' ref.className] ['.' property]

Tabla 2. Reglas de generación de código: de modelos ETL a código ETL

Así, la Tabla 2 muestra las relaciones existentes en la transformación de los modelos ETL a código ETL. En este caso, la descripción en lenguaje natural incluye algunos términos propios de la descripción de la gramática, como por ejemplo el uso de corchetes para describir elementos opcionales.

Para la implementación del generador de código ETL se ha utilizado el lenguaje MOFScript [22]. Se ha escogido este lenguaje porque ofrece soporte completo en Eclipse, existe gran cantidad de documentación y su curva de aprendizaje, desde nuestro punto de vista, es menor que para otras alternativas como xText [10] o Acceleo [21]. Además, ya fue utilizado satisfactoriamente en trabajos anteriores ([13], [29]).

Para la implementación del generador de código con MOFScript, se debe crear un fichero (extensión `.m2t`) donde se especifica el código que se generará a partir de cada elemento del metamodelo. Las transformaciones MOFScript tienen un comportamiento imperativo por lo que se debe definir qué regla se ejecutará en primer lugar (palabra reservada `main`). En este caso, la generación del código ETL comienza por la regla que genera el código correspondiente al elemento raíz del modelo ETL (`EtlModule`). El código que implementa esta regla es el siguiente:

```
eco.EtlModule::main () {
   var name:String
   if (self.name.size()=0)
    name="generated.etl"
   else
    name=self.name + ".etl"
   file (name)

   println("pre "+self.pre.name+" {")
    if (self.name.size()!=0)
    println("'Running ETL: "+self.name+
    " Transformation'.println();")
    self.pre.body println("}")

   println("post "+self.post.name+" {")
    self.post.body println("}")

   self.rules->forEach(r:eco.TransformationRule){
      r.generateRule()}

   self.operations->forEach(o:eco.Operation){
      o.generateOperation()}
}
```

Dado que se trata de la regla principal, esta regla debe definir dónde se almacenará el código resultante. Para ello, se emplea la función `file`. En este caso, se ha decidido que el fichero de código se guarde con el nombre del elemento raíz y la extensión propia de los ficheros de transformación ETL (`.etl`). En caso

de que no se haya definido el nombre del módulo de la transformación, el nombre por defecto del fichero de código obtenido será `generated.etl`.

Como se puede observar en el código anterior, para generar el código relativo a los bloques *pre* y *post*, dado que los atributos de la metaclase `EolBlock` (`name` y `body`) son de tipo `String`, tan solo es necesario incluir su nombre para añadirlos al código resultante. En el caso de las reglas y las operaciones, dado que son estructuras más complejas, se invocan las reglas `generateRule()` y `generateOperation()`, que generan el código correspondiente a estos elementos. A modo de ejemplo, a continuación se muestra el código que implementa la regla `generateOperation()`:

```
eco.Operation::generateOperation(){
  if(self.annotation!=null and self.annotation.size()>0)
   println("@"+self.annotation)
  if(self.pre!=null and self.pre.body.size()>0)
   println("@pre "+self.pre.body)
  if(self.post!=null and self.post.body.size()>0)
   println("@post "+self.post.body)
  print("operation ") self.context.generateOPstatement()
  print(" "+self.name+"(")
  self.parameters->forEach(p:eco.OperationParameter){
       p.generateOpParameter()
       if(p!=self.parameters.last())print(",")
  }
  print(") : ")
  self._getFeature("return").generateOPstatement() println(" {")
  println("\t"+self.body)
  println("}")
}
```

En esta regla, se genera el código de la operación que incluye: pre-condiciones y post-condiciones, cabecera y cuerpo de la operación. La cabecera, a su vez, se compone del contexto de la operación, el nombre, los parámetros y el tipo de retorno. Para facilitar la comprensión de esta regla, la Fig. 3 muestra el código generado para una operación ETL que puede invocarse sobre un elemento `Families!Father` (perteneciente al caso de estudio que se muestra en la siguiente sección) y devuelve un `String` que se forma a partir del valor de dos atributos del elemento de entrada.

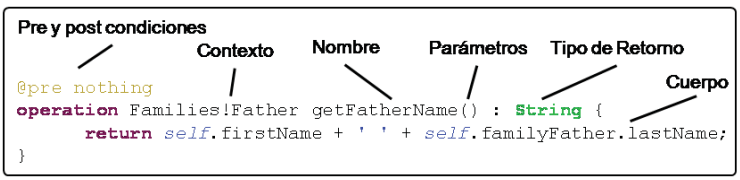


Figura 3. Código generado para una operación ETL

El resto de relaciones mostradas en la Tabla 2 se han implementado de forma similar, mediante reglas en MOFScript.

4. Caso de Estudio

Para ilustrar la propuesta presentada, en esta sección, se presenta un caso de estudio sencillo pero completo que es comúnmente empleado en el ámbito de las transformaciones de modelos: `Families2Persons` [1]. consiste en el desarrollo de una transformación para convertir modelos conformes al metamodelo `Families` (Fig. 4.a) en modelos conformes al metamodelo `Persons` (Fig. 4.b). Conviene mencionar que esta propuesta también ha sido utilizada en escenarios más complejos que pueden ser consultados en [13].

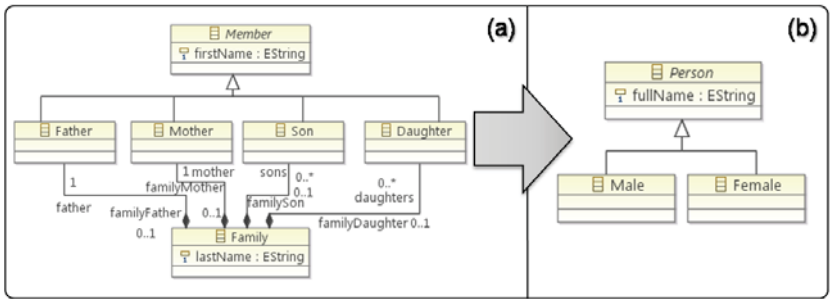


Figura 4. Metamodelos `Families` y `Persons`

Para desarrollar una transformación entre estos dos metamodelos usando la propuesta presentada, en primer lugar se debe modelar la transformación a alto nivel mediante un modelo de aproximación híbrida (`families2persons.hybrid`). Posteriormente, se debe ejecutar la transformación M2M presentada en la sección anterior que consumirá dicho modelo y producirá un modelo de transformación ETL (`families2persons.etl_model`). A partir del modelo ETL obtenido, se genera el código que implementa la transformación (`families2persons.etl`), mediante la ejecución del generador de código implementado con MOFScript.

La Fig. 5 muestra el desarrollo de la transformación `Families2Persons` usando la propuesta dirigida por modelos que se presenta en este trabajo. En concreto, la figura se centra en mostrar la regla `Mother_2_Female` a lo largo de los distintos niveles de abstracción en los que se ha definido la transformación.

5. Conclusiones y Trabajos Futuros

El desarrollo de transformaciones de modelos es una actividad inherentemente compleja. Dado que las transformaciones son otro artefacto software, sería conveniente también aplicar los principios de MDE [26] a su desarrollo. Así, el desarrollo de transformaciones de modelos podría beneficiarse de las ventajas

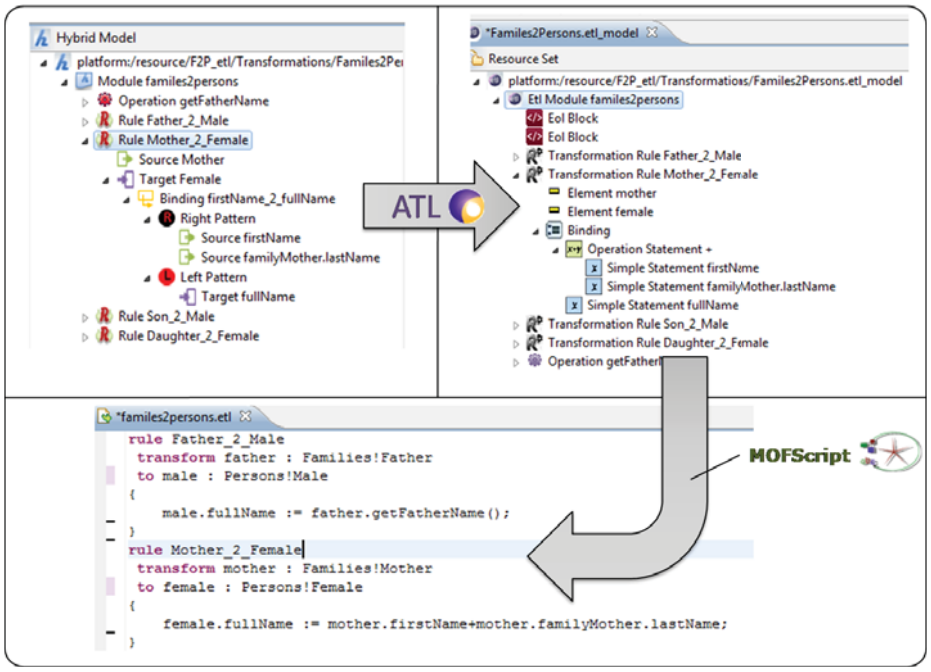


Figura 5. Caso de Estudio 'Families2Persons': de modelo de alto nivel a código ETL

que ofrece la aplicación de MDE: disminuir el nivel de complejidad, aumentar la productividad, facilitar la construcción de puentes tecnológicos entre diferentes tecnologías, etc.

Para poner en práctica esta idea, en este trabajo se ha presentado un proceso para el desarrollo dirigido por modelos de transformaciones de modelos en ETL [16], un lenguaje de transformaciones que adopta la aproximación híbrida (combinación de los estilos declarativo e imperativo). En concreto, se ha presentado la implementación de un metamodelo para dicho lenguaje, una transformación de modelo a modelo que permite generar modelos ETL a partir de especificaciones de alto nivel propias de la aproximación híbrida y una transformación de modelo a texto para la generación del código ETL que implementa la transformación modelada. Finalmente, para ilustrar el funcionamiento de la propuesta se ha presentado un caso de estudio en el que se ha seguido el proceso propuesto para el desarrollo de transformaciones de modelos.

El resultado de este trabajo proporciona diferentes líneas de trabajo futuras. Dado que en trabajos anteriores ([6], [12]) ya se mostró la validez de esta propuesta para el desarrollo de transformaciones para los lenguajes ATL y RubyTL, la línea más inmediata pasa por aplicar las mismas ideas para el desarrollo de transformaciones en otros lenguajes, como por ejemplo VIATRA [8]. Además, una vez que se disponga de los metamodelos de los diferentes lenguajes de transformación será posible construir puentes tecnológicos entre estos lenguajes, mejorando el nivel de interoperabilidad entre ellos. Por otra parte, al igual

que en este trabajo se han definido las transformaciones a nivel PSM siguiendo la aproximación híbrida, podrían definirse otros metamodelos a este nivel que sigan otras aproximaciones (puramente imperativa o declarativa, basada en grafos, etc.). Así, el desarrollador podría seleccionar la aproximación que quiere seguir y luego escoger entre los lenguajes que adopten dicha aproximación.

Agradecimientos

Este trabajo de investigación se ha llevado a cabo en el marco de trabajo del proyecto MASAI (TIN-2011-22617), financiado por el Ministerio de Ciencia e Innovación del Gobierno de España.

Referencias

1. Allilaire, F., Jouault, F. (2007): ATL Use Case - Families to Persons. ATLAS, INRIA and University of Nantes. `http://www.eclipse.org/m2m/atl/doc/ATLUseCase_Families2Persons.pdf`
2. Bernstein, P. (2003): Applying model management to classical meta data problems. First Biennial Conference on Innovative Data Systems Research, Asilomar, USA.
3. Bézivin, J., Büttner, F., Gogolla, M., Jouault, F., Kurtev, I., Lindow, A. (2006): Model Transformations? Transformation Models!. 9th International Conference on Model Driven Engineering Languages and Systems, MoDELS 2006.
4. Bohlen, M.: QVT und Multi-Metamodell-Transformationen in MDA. In: OBJEKTspektrum, vol. 2. Translated in: http://goo.gl/boRs4. (2006)
5. Bollati, V. A., Sánchez, E. V., Vela, B., Marcos, E. (2009): Análisis de QVT Operational Mappings: un caso de estudio. VI Taller sobre Desarrollo de Software Dirigido por Modelos (DSDM). Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos (JISBD2009). vol. 3 (2).
6. Bollati, V. A. (2011): MeTAGeM: un Entorno de Desarrollo de Transformaciones de Modelos Dirigido por Modelos. Tesis Doctoral, Universidad Rey Juan Carlos.
7. Budinsky, F., Merks, E., Steinberg, D. (2008). Eclipse Modeling Framework 2.0 (2nd Edition): Addison-Wesley Professional.
8. Csertán, G., Huszerl, G., Majzik, I., Pap, Z., Pataricza, A., Varró, D. (2002): VIATRA-visual automated transformations for formal verification and validation of UML models. 17th IEEE International Conference on Automated Software Engineering (ASE 2002), pp. 267-270.
9. Czarnecki, K., Helsen, S. (2003): Classification of model transformation approaches. OOPSLA'03, workshop on Generative Techniques in the Context of MDA.
10. Efftinge, S., Völter, M. (2006): oAW xText: a framework for textual DSLs. Workshop on Modeling Symposium at Eclipse Summit.
11. Guerra, E., de Lara, J., Kolovos, D., Paige, R., dos Santos, O. (2010): transML: A Family of Languages to Model Model Transformations. Model Driven Engineering Languages and Systems. vol. 6394, pp. 106-120, Springer.
12. Jiménez, Á., Granada, D., Bollati, V. A., Vara, J. M. (2011): Using ATL to support Model-Driven Development of RubyTL Model Transformations. 3rd International Workshop on Model Transformation with ATL (MtATL2011), Zürich, Switzerland.

13. Jiménez, Á. (2012): Incorporando la Gestión de la Trazabilidad en un entorno de Desarrollo de Transformaciones de Modelos Dirigido por Modelos. Tesis Doctoral, Universidad Rey Juan Carlos, Abril 2012.
14. Jouault, F., Kurtev, I. (2006): Transforming Models with ATL. Satellite Events at the MoDELS 2005 Conference. vol. 3844, Springer, pp. 128-138.
15. Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I. (2008): ATL: A model transformation tool. Science of Computer Programming, vol. 72 (1-2), pp. 31-39.
16. Kolovos, D. S., Paige, R. F., Polack, F. A. C. (2006): The Epsilon Object Language (EOL). Model Driven Architecture - Foundations and Applications. vol. 4066, Springer Berlin / Heidelberg, pp. 128-142.
17. Kolovos, D. S., Paige, R. F., Polack, F. A. C. (2008): The Epsilon Transformation Language. Theory and Practice of Model Transformations. vol. 5063, pp. 46-60, Springer Berlin / Heidelberg.
18. Kolovos, D. S., Rose, L. M., Paige, R. F., Polack, F. A. C. (2010): The Epsilon Book. `http://www.eclipse.org/epsilon/doc/book/`
19. Kusel, A. (2009): TROPIC - a framework for building reusable transformation components. Proceedings of the Doctoral Symposium at MODELS 2009, School of Computing, Queen's University, Denver, 2009.
20. Lano, K. Kolahdouz-Rahimi, S. (2011): Model-Driven Development of Model Transformations. Theory and Practice of Model Transformations. vol. 6707, pp. 47-61, Springer.
21. Musset, J., Juliot, E., Lacrampe, S. (2006): Acceleo User Guide. `http://www.acceleo.org/doc/obeo/en/acceleo-2.6-user-guide.pdf`.
22. Oldevik, J. (2006): MOFScript user guide, Version 0.6 (MOFScript v1.1. 11).`http://www.eclipse.org/gmt/mofscript/doc/MOFScript-User-Guide.pdf`
23. OMG (2003): MDA Guide Version 1.0.1 Document number omg/03-06-01. Ed.: Miller, J. and Mukerji, J. `http://omg.org/cgi-bin/doc?omg/03-06-01`.
24. Rose, L. M., Herrmannsdoerfer, M., Williams, J.R., Kolovos, D. S., Garcés, K., Paige, R. F., and Polack, F. (2010): A Comparison of Model Migration Tools. Proceedings of the 13th International Conference on Model Driven Engineering Languages and Systems (MODELS'10), Springer, Berlin, pp. 61-75.
25. Sánchez, J., García, J., Menarguez, M. (2006): RubyTL: A Practical, Extensible Transformation Language. European Conference on Model Driven Architecture - Foundations and Applications, ECMDA-FA.
26. Schmidt, D. C. (2006): Model-driven engineering. IEEE Computer, vol. 39 (2), pp. 25-31.
27. Stahl, T., Völter, M., and Czarnecki, K. (2006): Model-Driven Software Development: Technology, Engineering, Management. John Wiley & Sons.
28. van Amstel, M., van den Brand, M.: Using Metrics for Assessing the Quality of ATL Model Transformations. In: 3rd International Workshop on Model Transformation with ATL (MtATL2011), Zürich, Switzerland 2011, pp. 20-34. CEUR-WS
29. Vara, J. M. (2009): M2DAT: A Technical Solution for Model-Driven Development of Web Information Systems. Tesis Doctoral, Universidad Rey Juan Carlos, 2009.
30. Völter, M. (2009): Best Practices for DSLs and Model-Driven Development. Journal of Object Technology, vol. 8 (6), pp. 79 - 102, September-October 2009.
31. Xiong, Y., Liu, D., Hu, Z., Zhao, H., Takeichi, M., Mei, H. (2007): Towards automatic model synchronization from model transformations. Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering, Atlanta, USA, 2007, pp. 164-173.

Un proceso de modernización dirigido por modelos de sistemas web heredados hacia SOAs

Encarna Sosa, Pedro J. Clemente, José M. Conejero, and Roberto Rodríguez-Echeverría

Quercuss Software Engineering Group. Universidad de Extremadura
{esosa, pjclemente, chemacm, rre@unex.es} *

Resumen En estos últimos años, empresas y administraciones públicas han desarrollado un ecosistema de aplicaciones Web para ofrecer servicios en general, tanto hacia Internet como hacia sus intranets. Sin embargo, tanto empresas como administraciones públicas están descubriendo que sus webs no están alineadas con sus procesos de negocios, ya que éstas se crearon para solventar problemas concretos, en algunos casos duplicando funcionalidades y sin tener en cuenta la naturaleza cambiante de los procesos de negocio. La modernización de estas aplicaciones Web, utilizando una Arquitectura Orientada a Servicios (SOA) permitiría alinear la infraestructura tecnológica (webs) con los procesos de negocio. En este artículo se define un proceso sistemático y semi-automático de modernización arquitectónica del ecosistema de aplicaciones webs de una organización hacia una arquitectura orientada a servicios.
Keywords: *model-driven modernization; service-oriented architecture*

1 Introducción

Desde hace años está creciendo el número de servicios proporcionados a través de la Web, tanto por empresas como por administraciones públicas. Sin embargo, en muchos casos, este crecimiento de servicios se está realizando de forma rápida, incluso desorganizada, sin tener en cuenta los procesos de negocio de la organización. Así, los procesos de negocio no están adecuadamente alineados con la infraestructura tecnológica de la organización, especialmente, con las aplicaciones Web de la misma.

Dada esta situación, hay una tendencia actual para modernizar aplicaciones Web (WAs) tradicionales hacia Arquitecturas Orientadas a Servicios (SOA, en inglés) o Arquitecturas Orientadas a la Web (WOA, en inglés), cuyo principal objetivo es garantizar la interoperabilidad, flexibilidad y adaptabilidad de los sistemas, facilitando la alineación de los procesos de negocio con la infraestructura tecnológica [1]. Sin embargo, los procesos de modernización a menudo se llevan

* This work has been developed under the support of both, Spanish Contract MIGRARIA - TIN2011-27340 funded by Ministerio de Ciencia e Innovación, Gobierno de Extremadura and FEDER.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 471-476, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

a cabo de forma ad hoc, desarrollando las nuevas aplicaciones Web desde cero, resultando un proceso caro y propenso a errores. Por ello, la industria demanda la formalización y estandarización de los procesos de reingeniería de aplicaciones Web. Este artículo describe un proceso dirigido por modelos, sistemático y semi-automático, para modernizar WAs heredadas (aplicaciones desarrolladas en el pasado) hacia SOAs. Así, mientras que en un proceso de modernización tradicional el razonamiento sobre cómo modernizar las WAs se realizaría a nivel de código, la utilización de técnicas de DSDM permite razonar y procesar los modelos de las mismas desde un nivel de abstracción más elevado y no ligado a la tecnología concreta en las cuales estén desarrolladas las WAs.

Este trabajo es parte de un proyecto de investigación llamado MIGRARIA [2] donde, siguiendo la filosofía de ADM, se propone la modernización de aplicaciones Web no basadas en modelos y dirigidas a datos en Aplicaciones de Internet Enriquecidas (RIA). En este sentido, la fase de ingeniería inversa descrita en este artículo reutiliza el proyecto MIGRARIA.

Las principales contribuciones de este artículo son: la descripción del contexto donde se requiere la modernización de WAs heredadas en SOAs y la definición del proceso de modernización dirigido por modelos de WAs a SOA.

El resto del artículo está estructurado de la siguiente forma: en la sección 2 se presenta un ejemplo que ilustra la necesidad del proceso de modernización. En la sección 3 se describen las principales fases del proceso. Finalmente, en la sección 4 presentamos las conclusiones principales y trabajos futuros.

2 Ejemplo de la motivación

En esta sección describimos un ejemplo de la motivación para llevar a cabo este trabajo. Hemos seleccionado nuestra universidad, la Universidad de Extremadura (UEx), una organización en la que se justifica la necesidad de un proceso de modernización debido a sus características:

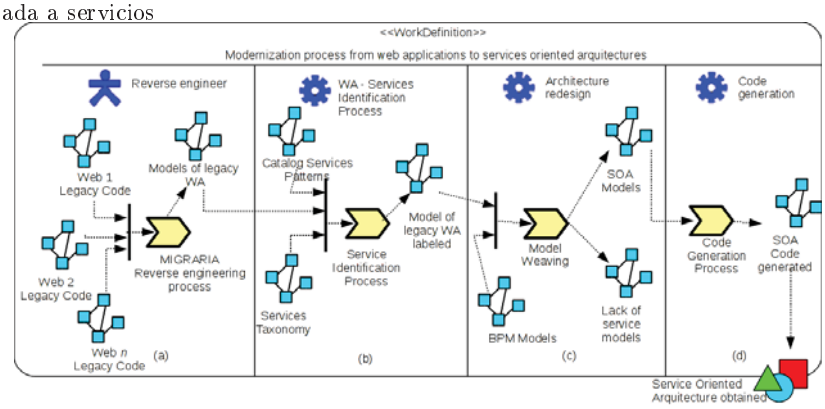
- Algunas WAs han sido desarrolladas para ofrecer múltiples servicios a estudiantes, profesores, personal de administración y público en general. Ejemplos de esas webs son aquellas que han sido creadas para servicios de estudiantes y profesores[1], investigación[2], instalaciones deportivas[3], asuntos académicos[4], evaluación de la calidad de la docencia[5], biblioteca[6], etc.
- Esas WAs fueron diseñadas para resolver problemas concretos, sin tener en cuenta una perspectiva general de los procesos de negocio universitarios. Como consecuencia, cada aplicación Web está normalmente aislada del resto y podemos encontrar servicios duplicados y/o que no intercambian datos o no son reutilizables.

[1] https://www.unex.es
[2] http://investigalia.unex.es
[3] http://158.49.134.24/CronosWeb/Modulos/Login.aspx
[4] https://academico.unex.es/actas/inicio.jsp
[5] http://uex21.unex.es:8002/app/docentia/AUTO/index.html
[6] http://biblioteca.unex.es

– Existe un gran número de webs diseñadas de las formas descritas, debido a ello, actualmente la UEx está revisando y documentando sus procesos de negocio para publicar su catálogo de servicios y, posteriormente, alinearlos con la infraestructura IT disponible.

Creemos que esta es una clara situación en la que podemos aprovechar el proceso que se describe en este artículo para reorganizar esas aplicaciones Web mediante la obtención de una aplicación Web organizada sobre una SOA.

Fig. 1. Visión general del proceso: desde aplicaciones Web hacia una arquitectura orientada a servicios

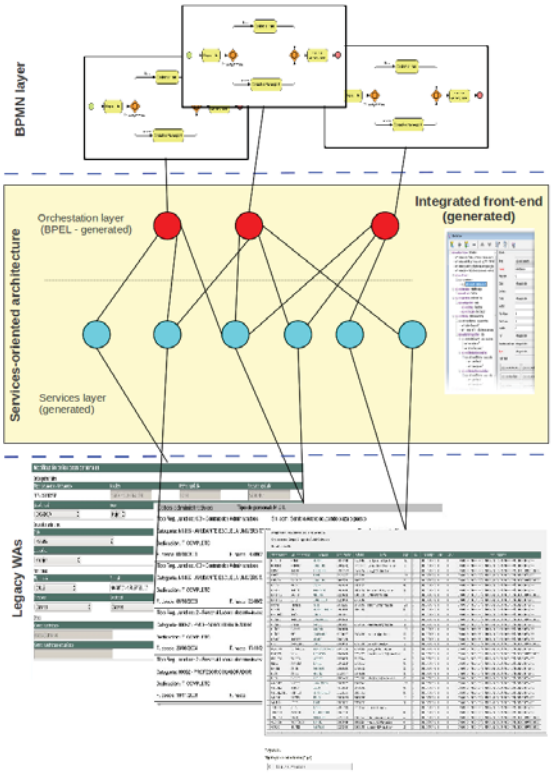


3 Visión general del proceso de modernización dirigido por modelos

En la Figura 1 se muestran las fases principales del proceso de modernización propuesto. El proceso incluye cuatro pasos: (a) ingeniería inversa; (b) identificación de servicios; (c) reestructuración de la arquitectura; (d) generación de código. En los siguientes párrafos se presenta una descripción más detallada de cada fase:

1. ***Ingeniería inversa***. Existen trabajos previos que utilizan ingeniería inversa para obtener una representación a nivel de modelos de los distintos servicios ofrecidos: mientras que en [3] se extraen los modelos a partir de la configuración de las aplicaciones basadas en el framework Struts, en [2] se obtiene el modelo de la aplicación a partir del código de las WAs heredadas basadas en MVC.
   Esta primera fase (identificada como *a* en la Figura 1) consiste en la extracción y representación de información del sistema heredado. Para ello,

Fig. 2. Arquitectura orientada a servicios obtenida en el proceso



reutilizaremos la fase de ingeniería inversa de aplicaciones Web propuesta en el proyecto MIGRARIA.

2. ***Identificación de servicios***. El principal objetivo de esta fase (*b* en la Figura 1) es la reordenación de los modelos de las aplicaciones Web para etiquetar los servicios que se proveen. Este proceso es crítico porque una adecuada identificación y catalogación de servicios permitirá identificar la red de servicios subyacente en las aplicaciones Web. Trabajos previos como [4] obtienen los procesos de negocio en BPMN a partir del código de aplicaciones heredadas; en nuestro caso, nos centraremos en procesar los modelos obtenidos en la fase 1 del proceso. Basándonos en estudios previos [5], [6], la catalogación de servicios se puede realizar teniendo como base una ontología (modelo de datos que representa un conjunto de conceptos de un dominio y sus relaciones) y una taxonomía de servicios (clasificación de los distintos tipos de servicios en categorías). Nuestro trabajo es un trabajo emergente que está actualmente en vías de desarrollo, por ello, nos gustaría aclarar que actualmente estamos trabajando en la fase de catalogación de servicios. Esta catalogación nos ayudará a identificar las propiedades y características co-

munes de los servicios que pertenecen a una categoría particular (propósito principal, comportamiento, interface, seguridad, etc.). Para ello, en esta fase se definirán dos artefactos adicionales para llevar a cabo la identificación de los servicios en los modelos WAs: una taxonomía de servicios y un catálogo de patrones de diseño definidos sobre servicios (patrones de autentificación, mantenimiento de sesiones, sincronización de datos, etc.). El primero (taxonomía de servicios) se utilizará para clasificar los servicios ofrecidos por las WAs de acuerdo con los tipos de servicios existentes en SOAs. Para este propósito, se definen las categorías de servicios y las relaciones entre ellas. El segundo artefacto (catálogo de patrones de diseño de servicios) [7] se utilizará para buscar en los modelos de WA ocurrencias de esos patrones con el objetivo de ayudar al arquitecto software a etiquetar servicios. La salida de esta fase será un modelo WA etiquetado, donde los servicios potenciales de la WA se han identificado modelándolos, por ejemlo, mediante SOAML, y catalogado. Además, este proceso de catalogación nos ayudará a detectar otras características importantes como similitud entre servicios, servicios que pueden estar duplicados, dependencias entre servicios, etc.

3. ***Reestructuración de la arquitectura a nivel de modelo***. El objetivo de esta fase (c en la Figura 1) es hacer un matching para enlazar los servicios identificados en la etapa anterior del proceso (fase 2) con los procesos de negocio definidos por la organización. Para ello, se lleva a cabo un proceso de tejido de modelos, que tiene dos entradas diferentes: por un lado, el modelo WA etiquetado (obtenido en la fase 2) y, por otro lado, los modelos de proceso de negocio de la empresa. En esta fase del trabajo partimos de la base de que los procesos de negocio son proporcionados por la organización (UEx en nuestro caso) y estarán definidos utilizando notaciones conocidas, como BPMN o similares [8].

   A la hora de enlazar servicios etiquetados con los procesos de negocio definidos podemos encontrar distintas situaciones, entre las que destacan:

   - Servicios que pueden unirse de una forma prácticamente directa (por ejemplo, servicios con la misma interfaz).
   - Servicios que deben ser adaptados mediante *wrappers* para alinearse con procesos BPM.
   - Servicios huérfanos en los modelos de las WAs o servicios que no han sido identificados correctamente en las fases anteriores.
   - Servicios que, potencialmente, pueden ser extendidos con funcionales ofrecidas por terceros (por ejemplo, servicios ofrecidos por redes sociales).

   El resultado de esta fase es el modelo formado por el tejido de los procesos BPM y los servicios etiquetados.

4. ***Generación de código***. La fase de generación de código (d en la Figura 1) no es una contribución novedosa, en ella utilizaremos distintas técnicas [9] para generar los *wrappers* necesarios y reestructurar el acceso a los servicios. Así, además de ofrecer los servicios de forma interoperable (ver Figura 2), bien mediante Servicios Web o bien mediante servicios REST, se generará una capa de orquestación de servicios basada en BPEL.

4 Conclusiones y trabajos futuros

Actualmente hay un amplio ecosistema de aplicaciones web susceptibles de ser modernizadas para obtener SOAs, como evidencia el ejemplo de la motivación descrito, que puede ser extrapolado a otras organizaciones. Este amplio número de WAs heredadas muestra la necesidad de procesos sistemáticos de modernización de WAs.

En este artículo se ha descrito un proceso sistemático y semi-automático dirigido por modelos para modernizar WAs heredadas hacia SOAs. Mediante este proceso, los servicios que ofrecen las WAs heredadas pueden ser reutilizados en lugar de ser re-implementados, mejorando así su reusabilidad. El proceso propone un mecanismo basado en el tejido de modelos para alinear los servicios que subyacen en las WAs heredadas con los procesos de negocio de la empresa.

Como hemos comentado previamente, este trabajo está en su fase inicial de desarrollo, por tanto, las líneas de investigación que se van a llevar a cabo durante su desarrollo completo incluyen: clasificación de servicios, patrones de diseño orientados a servicios, extracción e identificación de servicios desde los modelos WA, ingeniería inversa aplicada a WAs, model weaving – aplicado a WA y modelos BPMN – y generación de código (para envoltorios, código específico de plataforma, implantación de artefactos, etc.).

References

1. Bell, M.: Service-Oriented Modeling: Service Analysis, Design, and Architecture. Wiley Publishing (2008)
2. Rodríguez-Echeverría, R., Conejero, J.M., Clemente, P.J., Preciado, J.C., Sánchez-Figueroa, F.: Modernization of Legacy Web Applications into Rich Internet Applications. In Harth, A., Koch, N., eds.: 7th Model-Driven Web Engineering Workshop. Volume 7059/2012., Paphos, Cyprus, Lecture Notes in Computer Science. Springer Verlag. (2011) 236–250
3. Izquierdo, J.L.C., Molina, J.G.: A domain specific language for extracting models in software modernization. In Paige, R.F., Hartman, A., Rensink, A., eds.: ECMDA-FA. Volume 5562 of Lecture Notes in Computer Science., Springer (2009) 82–97
4. Pérez-Castillo, R., de Guzmán, I.G.R., Piattini, M.: Business process archeology using MARBLE. Information & Software Technology **53**(10) (2011) 1023–1044
5. Cohen, S.: Ontology and taxonomy of services in a service-oriented architecture. http://msdn.microsoft.com/en-us/library/bb491121.aspx (April 2007)
6. Fareghzadeh, N.: Service identification approach to soa development. In: Proceedings of World Academy of Science, Engineering and Technology. (2008) 258–266
7. Vora, P.: Web Application Design Patterns. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2009)
8. Ko, R.K., Lee, S.S., Lee, E.W.: Business process management (bpm) standards: a survey. Business Process Management Journal **15**(5) (2009) 744–791
9. Zhang, Z., Yang, H.: Incubating services in legacy systems for architectural migration. In: APSEC, IEEE Computer Society (2004) 196–203

Un framework basado en modelos para la modernización de datos

Fco. Javier Bermúdez Ruiz and Jesús García Molina

Universidad de Murcia,
Murcia, España
{fjavier,jmolina}@um.es

Abstract. *La modernización de software permite a las empresas mantener el valor estratégico de sus sistemas legacy (legados). La reingeniería de datos es un tipo de modernización que mejora la calidad de los datos de dichos sistemas. En este trabajo presentamos una arquitectura basada en las técnicas de la Ingeniería de Software Dirigida por Modelos que automatiza la tarea de la reingeniería de los datos relacionada con la mejora de la calidad y la migración de datos. Además la solución propuesta también proporciona independencia respecto del sistema gestor de base de datos a modernizar.*

Keywords: Ingeniería de Software Dirigida por Modelos, Modernización de datos, Reingeniería de datos

1 Introducción

Las técnicas y aplicaciones de la Ingeniería de Software Dirigida por Modelos (Model-Driven Engineering, MDE) han despertado un gran interés tanto en la comunidad científica como en la industria por su capacidad para dominar la complejidad del software y mejorar su calidad. En MDE, el uso sistemático de los modelos permite elevar el nivel de abstracción e incrementar la automatización en el desarrollo de software. Aunque las aplicaciones más conocidas de MDE están destinadas a la creación de nuevo software (MDA, factorías de software, desarrollo específico del dominio) también pueden ser aplicadas con éxito a la modernización de software [1][2].

Cuando se requieren cambios de más alcance que un simple mantenimiento, un sistema legacy debe ser modernizado para mantener su valor para la empresa. La reingeniería es un tipo especial de modernización de software en la que se mejora la calidad de un sistema según un método sistemático [3]. En este trabajo nos centramos en el proceso de reingeniería de datos (*data reengineering*) que se ocupa de derivar una nueva base de datos por causas como la necesidad de migrar a una nueva plataforma o para mejorar la calidad de los datos almacenados [4]. La reingeniería de datos es un área bien conocida para la que existen fundamentos teóricos, metodologías, herramientas y experiencia en la industria, en la que las técnicas MDE pueden ser aplicadas para mejorar la automatización [5].

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 477-490, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

Un problema frecuente con bases de datos legacy es la carencia de restricciones estructurales (implícitas en los datos) y de un nivel de normalización adecuado. En la actualidad no hay muchos trabajos que aborden la automatización del proceso de normalización y suele ser un proceso manual y consecuentemente poco productivo. En este trabajo se presenta un enfoque MDE para aplicar mejoras de la calidad de las bases de datos y que permite resolver, entre otros, problemas como los mencionados. A través de una arquitectura que involucra varias cadenas de transformaciones de modelos se ofrece un soporte para elevar el nivel de automatización en escenarios de reingeniería de datos, como la calidad de datos o la migración a una nueva plataforma. Además, la aplicación de técnicas MDE permite que el proceso sea independiente tanto de los sistemas de gestión de bases de datos utilizados como del esquema de datos sobre el que se aplica la reingeniería. Se ha implementado un prototipo del framework en el marco de un proyecto CDTI[1] de automatización de migración de aplicaciones Oracle Forms.

El artículo se organiza de la siguiente forma: en la Sección 2 se presenta la motivación del trabajo y se resume el proceso de migración de datos, mostrando el caso de estudio que guía el artículo; en las Secciones 3, 4 y 5 se describen las etapas del proceso: identificación de errores, corrección de errores y generación de código, respectivamente; en la Sección 6 se presenta el trabajo relacionado; por último, en la Sección 7 se indican las conclusiones y el trabajo futuro.

2 Motivación y proceso de modernización

Nuestro grupo de investigación ha colaborado con la empresa Sinergia Tecnológica en el marco de un proyecto CDTI cuyo objetivo era automatizar la migración de aplicaciones Oracle Forms a la plataforma Java. Se diseñó una estrategia de migración dirigida por modelos para conseguir automatizar algunas tareas e independizar la migración de las tecnologías subyacentes. En este trabajo nos centraremos en la arquitectura definida para la migración de la capa de datos, en la que no sólo nos preocupamos de generar el código para la plataforma destino (p.e. JPA [6]) sino que también afrontamos la mejora de la calidad de la base de datos de la aplicación a migrar. En una migración de datos es frecuente encontrar que la estructura de la base de datos presenta deficiencias tanto en su forma normal como en sus características de integridad (p.e. la ausencia de clave ajena), por lo que es deseable su mejora.

La Figura 1 muestra un esquema global del proceso de migración de datos definido en este trabajo. A un esquema relacional inicial se le aplica un conjunto de transformaciones con el propósito de eliminar deficiencias importantes. Sobre el esquema mejorado resultante se aplica una normalización a partir de la cual se obtendrán las clases Java de acceso a la base de datos para un gestor de persistencia como JPA.

[1] Centro para el Desarrollo Tecnológico Industrial (CDTI). Proyecto Num. IDI-20100500: Herramienta Orientada a la Migración Basada en Modelos. Ministerio de Economía y Competitividad. 2010-2011. http://www.cdti.es.

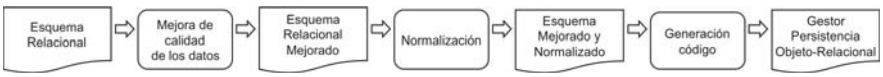


Fig. 1. Descripción del problema

En el proyecto de migración se han considerado las siguientes deficiencias: (a) si se detectan patrones de repetición de los datos de varias columnas, con respecto a las columnas que son clave principal de otra tabla, se infiere una clave ajena (foreign key, FK); (b) si se detectan claves ajenas y restricciones de columnas (check constraints, CK) creadas sobre una tabla, pero que por alguna razón (datos históricos, migraciones, nuevos requisitos, etc.) se han desactivado, se reactivan. Las anteriores deficiencias son sólo una muestra de las que pueden ser consideradas.

Una migración es un proceso de reingeniería que consta de tres etapas [7] como muestra la Figura 2. Una primera etapa de *ingeniería inversa* se encarga de extraer conocimiento sobre la aplicación a migrar a diferentes niveles de abstracción y de transformarlo para obtener nuevo conocimiento a diferentes niveles de abstracción. En el caso de una migración dirigida por modelos, se usan metamodelos para representar el conocimiento y una cadena de transformaciones modelo-a-modelo (m2m) para obtener nuevo conocimiento a partir del extraído directamente del sistema. Esta cadena de transformación debe arrancar con una transformación texto-a-modelo (t2m) que extraiga o inyecte un modelo a partir del código disponible de la aplicación. La segunda etapa consiste en una *reestructuración* que transforma las representaciones obtenidas en la ingeniería inversa en otras conformes al sistema destino, por ejemplo una transformación de la arquitectura del sistema. En nuestro caso, se ha implementado también mediante una cadena de transformaciones m2m . Finalmente, una etapa de *ingeniería directa* genera el nuevo sistema. En el caso de una migración dirigida por modelos, esta etapa se implementa mediante una cadena de transformaciones de modelos, cuya última etapa es una transformación modelo-a-texto (m2t) que genera artefactos del nuevo sistema.

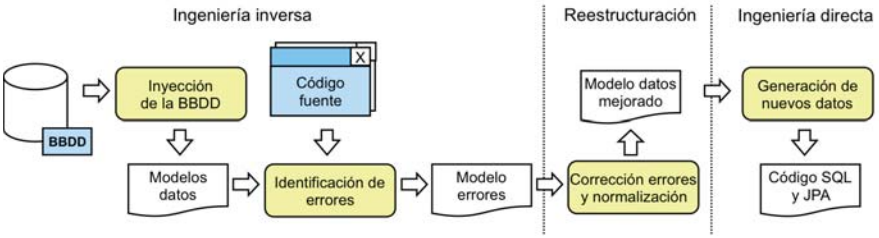


Fig. 2. Proceso de modernización de datos

La Figura 2 muestra el proceso de migración de datos que se ha aplicado, el cual se ajusta a las etapas indicadas anteriormente pero para una reingeniería de datos. A continuación se indican los pasos de cada una de estas etapas, las cuales serán descritas en las siguientes secciones: (1) se infieren y representan en un modelo de errores las posibles restricciones del esquema original (*Ingeniería Inversa*), requiriendo de una tarea previa de inyección de la base de datos; (2) se aplica la corrección de errores detectados sobre el esquema original y la normalización para obtener un esquema corregido y normalizado (*Reestructuración*), (3) por último se genera el código para la migración a la nueva plataforma a partir del esquema corregido (*Ingeniería Directa*). Se ha creado un framework basado en modelos que soporta estas tres etapas.

A continuación se muestra el ejemplo que se utilizará en el resto del artículo para ilustrar la aplicación del proceso de modernización. El esquema origen describe el típico modelo de datos para aplicaciones de comercio electrónico.

```
Tabla Cliente (idCliente, nombre, ciudad, provincia) PK<idCliente>
Tabla Venta (idVenta, fecha, idCliente) PK<idVenta>
Tabla LineaVenta (idLinea, idVenta, cantidad, idProducto)
  PK<idLinea,idVenta>
  FK<idVenta->Venta(idVenta)>
Tabla Producto (idProducto, nombre, precio, disponibilidad)
  PK<idProducto>
  CK<disponibilidad in ‘disponible’, ‘agotado’, ‘pedido’> disabled
```

3 Identificación de errores

La etapa de ingeniería inversa se ocupa de la identificación de errores y requiere del paso previo de inyección de los modelos.

3.1 Inyección

En la inyección de la base de datos se generan los modelos de datos (ver Figura 3). El modelo del esquema se extrae a partir de un script DDL/SQL. El modelo de los datos almacenados, si es necesario inyectarlos, se extrae a partir de un script DML/SQL. Se han creado los metamodelos DDL y DML que representan las sentencias de estos lenguajes y que no se muestran por razones de espacio. Sólo se ha considerado la información lógica de las tablas (columnas y tipos, clave primaria -primary key, PK-, claves ajenas y restricciones de columna) pero no la información física (tablespaces) ni otros elementos del esquema de datos (vistas, índices, secuencias, procedimientos, etc.). Como herramienta de inyección se usa Gra2MoL [8], un lenguaje que permite expresar transformaciones t2m como mappings entre elementos de una gramática y elementos de un metamodelo destino.

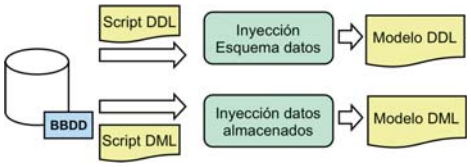


Fig. 3. Etapa de Inyección de la base de datos

3.2 Estrategias de Identificación

Se han desarrollado tres enfoques centrados en la identificación de posibles claves ajenas y restricciones de columna. Un enfoque identifica los errores explorando los datos almacenados mediante un modelo. Otro enfoque explora los datos directamente sobre el SGBDR. El tercer enfoque identifica los errores mediante un análisis del código de las aplicaciones cliente de la base de datos. Los tres enfoques se muestran en la Figura 4 . En esta etapa se detectan posibles mejoras relacionadas con problemas de integridad de datos causados por: (i) la ausencia de clave ajena, (ii) la existencia de clave ajena y restricción de columna desactivadas por migración de los datos desde una aplicación anterior, por necesidades de los datos o por cualquier otra causa.

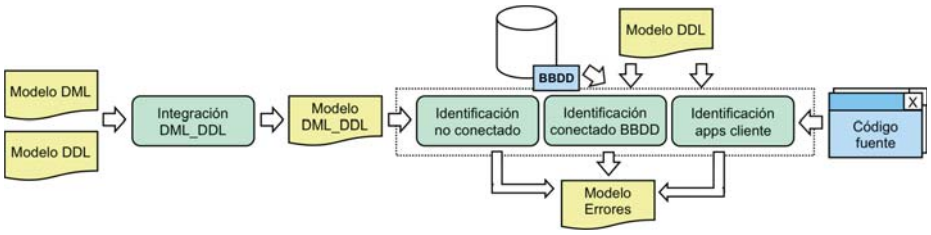


Fig. 4. Etapa de Identificación de errores

Los errores o posibles mejoras se representan con un modelo de errores que conforma al metamodelo mostrado en la Figura 5. Este metamodelo incluye una metaclase abstracta `Error` con subclases para cada tipo de error identificado: `ForeignError` y `CheckError`. El primero requiere información relativa a la tabla contenedora de la clave ajena, la tabla referenciada y las columnas implicadas. El segundo requiere las columnas y los valores de la restricción. Conviene aclarar que aunque se denomina modelo de errores, la información puede referirse a mejoras sobre la base de datos y no solo deficiencias.

La *identificación no conectada a base de datos* identifica los errores explorando los datos mediante modelos. Se ha definido un algoritmo que explora tanto el modelo DDL para detectar si existe alguna clave ajena y/o restricción de columna desactivada como el modelo DML para inferir posibles claves ajenas entre tablas. Dado que las mejoras consideradas son simples, el enfoque consiste

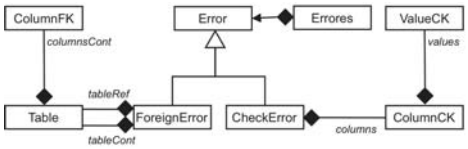


Fig. 5. Metamodelo de Errores simplificado

sencillamente en la asociación (*matching*) de valores de pares de columnas de distintas tablas, donde al menos una de las columnas es la clave primaria de su tabla y ambas son del mismo tipo. Además, debe haber alguna coincidencia parcial entre los nombres de ambas columnas y el matching de sus valores debe superar un umbral (es decir, porcentaje) de coincidencia.

En esta implementación se emplean los modelos DDL y DML obtenidos en la primera etapa de inyección. Ambos modelos se integran en un único modelo DML_DDL que se obtiene mediante una transformación m2m implementada con RubyTL [9]. Esta integración facilita toda la manipulación posterior para el algoritmo. El uso de modelos para los datos almacenados tiene como ventaja la independencia respecto del SGBDR y del lenguaje para la definición de los datos, pero como mayor inconveniente requiere el manejo de un modelo excesivamente grande. No obstante, el tamaño del modelo puede ser reducido considerando tan solo un extracto de datos significativo y manejable.

La *identificación conectada a base de datos* identifica los errores mediante la exploración directa de los datos almacenados en el SGBDR. El enfoque se ha implementado en PL/SQL y accede directamente al diccionario de datos del SGBDR del proveedor correspondiente (p.e. Oracle). Este enfoque permite tener un acceso rápido y completo de todos los datos almacenados, pero se trata de una solución dependiente del SGBDR origen.

En el caso del ejemplo introducido en la sección 2, en el proceso de identificación de errores se detectarían registros de las tablas `Cliente` y `Venta` como los siguientes:

```
Cliente (idCliente, nombre, ciudad, provincia) :
   [LR2, Luis, Elche, Alicante], [DC3, Diego, Oviedo, Asturias],
   [SA2, Sergio, Elche, Alicante], [PS5, Pedro, Elche, Alicante],
   [JM1, Juan, Valencia, Valencia]
Venta (idVenta, fecha, idCliente) :
   [V1, 20/12/2011, LR2], [V2, 01/05/2009, JM1],
   [V3, 05/09/2010, LR2], [V4, 19/05/2009, DC3],
   [V5, 20/06/2011, unknown],[V6, 18/12/2010, DC3]
```

Por asociación de valores entre columnas con igual tipo de distintas tablas, el par de columnas `<Cliente(idCliente),Venta(idCliente)>` tienen un 83% de coincidencia (superando el umbral establecido) por lo que se propone una clave ajena en `Venta` (tabla cuya columna no es clave primaria) entre ambas columnas. Por otro lado, las tablas `LineaVenta` y `Producto` contienen los registros:

```
Producto (idProducto, nombre, precio, disponibilidad)
   [P1, LCD, 120, agotado], [P2, Impresora, 150, disponible],
   [P3, LED, 175, disponible], [P4, Disco Duro, 90, pedido]
LineaVenta (idLinea, idVenta, cantidad, idProducto)
   [LV1, V1, 2, P2], [LV2, V1, 1, P4], [LV1, V2, 3, P4],
   [LV2, V2, 1, P1], [LV3, V2, 1, P3]
```

Igual sucede entre `<LineaVenta(idProducto),Producto(idProducto)>`, donde el porcentaje de coincidencia es del 100%. Además, se encuentra una restricción de columna desactivada en `Producto` y se comprueba que la activación obtiene un 100% de coincidencia de valores con la restricción.

Por último, la *identificación extraída de aplicaciones* identifica los errores mediante un análisis del código empleando técnicas de búsqueda de patrones en texto [10]. El análisis consiste en detectar relaciones entre columnas de distintas tablas declaradas en el código de las aplicaciones donde al menos una columna involucrada en la relación es primary key de su tabla, y descarta el uso de los datos almacenados. Dos columnas de distintas tablas están relacionadas si aparecen en una JOIN de la cláusula WHERE de alguna consulta SELECT en el código de la aplicación. Una vez detectadas las relaciones se procede a validar cuales están ya declaradas sobre el modelo DDL como clave ajena y aquellas que no lo están se proponen como posibles mejoras en el modelo de errores.

La implementación es dependiente de la plataforma de programación origen (Java, .Net, Oracle Forms, etc.) e incluso de su tecnología de acceso a datos. En este trabajo se ha implementado para la tecnología JDBC de Java. Para detectar las relaciones se requieren dos pasos: (1) inyectar sobre un modelo los fragmentos de sentencias SQL programadas en el código, (2) analizar el modelo de fragmentos y reconstruir las consultas para detectar las relaciones en la cláusula WHERE. A pesar de que la solución propuesta es dependiente de la tecnlogía de implementación, constituye una alternativa más fiable que la exploración de los datos almacenados, ya que mientras que con el uso de los datos se infieren relaciones por coincidencia de valores, con el código se detectan relaciones ya declaradas entre tablas.

En relación al caso de estudio, la identificación basada en código de aplicaciones cliente encuentra y analiza el siguiente fragmento de código:

```
Statement stm = con.createStatement();
ResultSet rsl = stm.executeQuery(‘‘SELECT v.idVenta, c.nombre
   FROM Venta v, Cliente c WHERE v.idCliente = c.idCliente’’);
```

De donde obtiene la relación `<Venta(idCliente),Cliente(idCliente)>` como posible clave ajena. El modelo de errores obtenido tras la etapa de identificación se muestra en la Figura 6.

Es conveniente que los resultados obtenidos durante la identificación de errores sean confirmados por parte del usuario del proceso de modernización. Por ello, se ha implementado un asistente para la visualización y confirmación de los errores identificados y descartar las propuestas que el usuario estime.

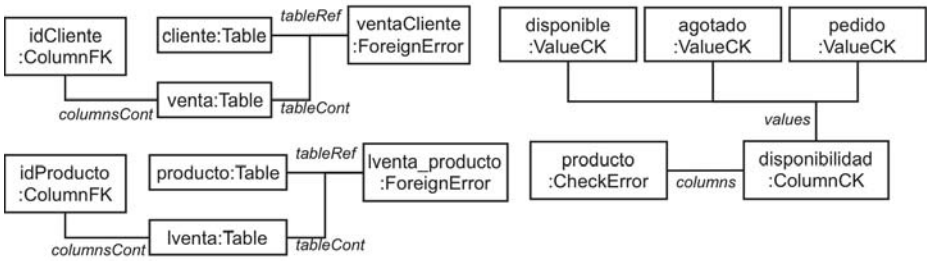


Fig. 6. Modelo de errores del caso de estudio

4 Corrección de errores

En esta sección se describirá cómo se implementa la etapa de reestructuración para el nuevo sistema (Figura 7). Esta etapa realiza la corrección de errores mediante una transformación m2m en RubyTL que obtiene un modelo DML_DDL correcto a partir del modelo de errores y del modelo DML_DDL original. Un modelo DML_DDL residual recoge los datos que no cumplen las restricciones incorporadas en la corrección.

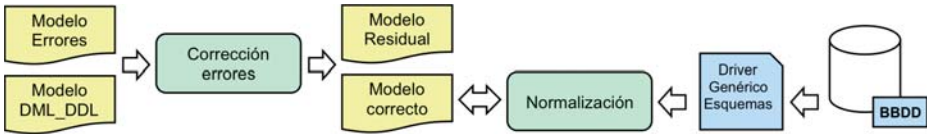


Fig. 7. Etapa de Corrección de Errores

Tras la corrección de errores se puede ejecutar la tarea de Normalización sobre el modelo corregido, la cual requiere varios pasos (ver Figura 8). Un primer paso es la identificación de las dependencias funcionales [11] que muestran las redundancias e inconsistencias de los datos; un segundo paso permite descomponer las tablas para eliminar las dependencias funcionales, siguiendo un enfoque de descomposición o de síntesis, según viabilidad.

Para la identificación de las dependencias funcionales se ha usado la herramienta ConceptExplorer [12] (ConExp) basada en los principios de FCA (Formal Concept Analysis) [13]. En [14] se comenta que FCA ha demostrado ser muy útil para representar el conocimiento almacenado en una base de datos y, en general, para calcular información contenida en ella. Este método toma como entrada una matriz de datos que representa (a) un grupo de objetos que comparten un subconjunto de propiedades denominado *cluster de objetos natural* y (b) un grupo de propiedades que reúne todos los atributos compartidos por dichos objetos, denominado *cluster de propiedades naturales*. En este ámbito, un *concepto* es un par formado por un cluster de objetos y su correspondiente clus-

ter de propiedades. Se define entonces un *contexto formal* como la agrupación de conceptos formado por un conjunto de objetos, un conjunto de propiedades y una indicación de qué objetos tienen qué propiedades. Para poder detectar las dependencias funcionales de las tablas usando FCA [14], ConExp recibe el contexto formal de cada tabla (en un formato XML propietario) y calcula el conjunto de implicaciones que se pueden deducir de dicho contexto. Estas implicaciones se corresponden con las dependencias funcionales presentes en la tabla y se inyectan en un modelo de dependencias funcionales. Para suministrar el contexto formal a ConExp se requiere el conjunto de objetos de cada tabla (los datos almacenados). Se ha implementado un driver de acceso genérico a esquemas para ofrecer un mecanismo de acceso uniforme a cualquier esquema de base de datos. Esto permite implementar un algoritmo genérico de exploración de tablas, columnas y registros con independencia del esquema. No se discute aquí por razones de espacio ya que se ha preferido ofrecer una visión global del framework.

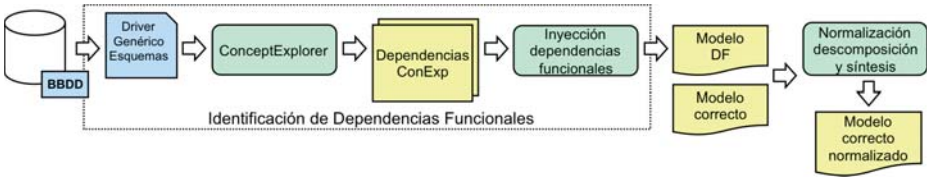


Fig. 8. Tarea de Normalización

Una vez se han identificado las dependencias funcionales se procede a la normalización del esquema de datos. El algoritmo de normalización consta de dos fases: una que realiza el paso de un modelo desnormalizado a un modelo a primera forma normal (1NF) y otra que realiza el paso de un modelo en 1NF a un modelo normalizado, en tercera forma normal (3NF) o forma normal de Boyce-Codd (BCNF) [11]. Para llevar a cabo este último paso existen dos alternativas [11]: el método de descomposición (o análisis) o el método de síntesis. Si bien ambos enfoques son válidos se considera a la descomposición como la alternativa principal, pues produce una normalización más precisa que la síntesis, ya que genera un esquema estructuralmente más parecido al esquema de partida. Por tanto, sólo se emplea la síntesis en aquellos casos en los que la descomposición sea insuficiente. El enfoque basado en descomposición parte del esquema original y descompone las tablas contenidas según las dependencias funcionales. El enfoque basado en síntesis parte de las dependencias funcionales por lo que la estructura obtenida suele diferir del esquema original en mayor medida que el primer enfoque. El algoritmo de descomposición se implementa con un backtracking, que descompone cada tabla según las dependencias funcionales contenidas en la misma; si se llegara a un punto en el cual no se pueda seguir descomponiendo y todas las dependencias funcionales de la tabla no han sido eliminadas, el algoritmo de backtracking retorna al estado de partida y se procede a aplicar normalización por síntesis para dicha tabla.

En relación al caso de estudio, para la normalización, primero se identifican las dependencias funcionales de cada tabla con ConExp y se obtiene la dependencia funcional `Tabla Cliente (ciudad->provincia)` para los registros de la tabla `Cliente` mostrados anteriormente. Tras la identificación se aplica el algoritmo de descomposición sobre `Cliente` para eliminar dicha dependencia funcional y se obtienen dos tablas en FNBC:

```
Cliente (idCliente, nombre, ciudad), Ciudad (ciudad, provincia)
```

Nótese que el administrador de la base de datos podría considerar aplicar denormalización por cuestiones de rendimiento.

5 Generación de nuevos datos

La última etapa genera el código de los artefactos resultantes de la modernización de la base de datos. Como se puede ver en la Figura 9, hay dos posibles alternativas: recrear la base de datos o generar código para el acceso a datos. Se puede optar por recrear la base de datos con los datos que se adecuan al nuevo esquema mediante los scripts DDL y DML. En ocasiones las empresas que necesitan modernizar los datos de su aplicación, requieren poder acceder al esquema relacional directamente (desde aplicaciones cliente, p.e. Oracle Forms o directamente desde procedimientos de base de datos). Esta alternativa permite a las empresas un control preciso sobre las estructuras generadas en el SGBDR. Otra alternativa consiste en generar una representación del esquema corregido y mejorado mediante un middleware de acceso a datos como JPA. Con este enfoque generativo las empresas obtienen un mayor nivel de abstracción en el manejo de los datos y mayores facilidades de uso. Sin embargo delegan en las implementaciones del middleware para representar la base de datos. Para la primera alternativa sólo se requiere una transformación m2t implementada en RubyTL. Para la alternativa JPA se ha diseñado una cadena de dos transformaciones: (1) transformación del modelo de base de datos mejorado en un nuevo modelo JPA usando una transformación m2m RubyTL y (2) obtención del código fuente a partir del modelo JPA usando una transformación m2t MOFScript [15]. Aunque se podría haber hecho mediante una transformación m2t directa, un paso intermedio simplifica la transformación al crear un modelo conforme a un metamodelo (Figura 10) que representa los elementos básicos de la arquitectura destino (JPA): unidad de persistencia, entidades y relaciones (entre otros).

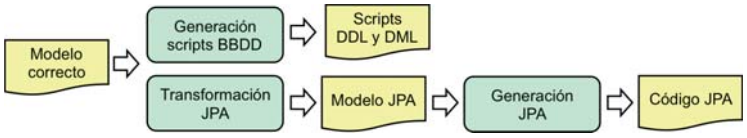


Fig. 9. Etapa de Generación de nuevos datos

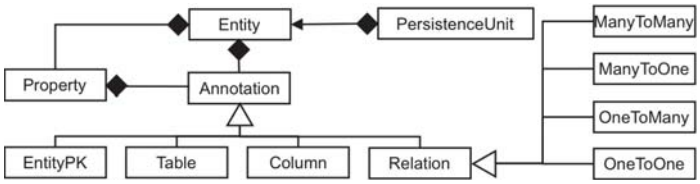


Fig. 10. Metamodelo JPA simplificado

Esta transformación m2m requiere la identificación de los distintos tipos posibles de relaciones entre tablas a partir de la información del modelo relacional. La transformación m2t genera una clase entidad por cada tabla, con atributos y sus métodos get() y set() por cada columna, y anotaciones para el identificador de la clase y las relaciones identificadas (uno-a-uno, uno-a-muchos, muchos-a-uno).

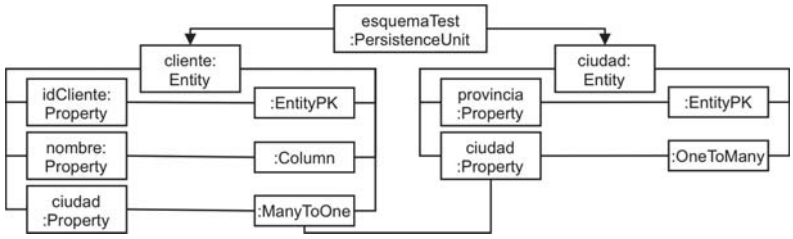


Fig. 11. Modelo JPA simplificado del caso de estudio para `Cliente` y `Ciudad`

En relación al caso de estudio, la Figura 11 muestra el modelo JPA obtenido en la última etapa de generación de código. Durante la generación de código se obtienen los siguientes artefactos: (1) fichero configuración `persistence.xml`, (2) clases anotadas `Cliente.java`, `Ciudad.java`, `Venta.java`, `LineaVenta.java` y `Producto.java`. Como ejemplo de entidad JPA generada se muestra parcialmente la clase `Ciudad`:

```
@Entity @Table(schema="TEST")
public class Ciudad implements Serializable {
        @Id @Column(name="CIUDAD")
        private String ciudad; ...
        @OneToMany(mappedBy="ciudad")
        private List<Cliente> clientes; ...
        public List<Cliente> getClientes() {
                return this.clientes; } ...
}
```

6 Trabajo relacionado

Nuestro trabajo se enmarca dentro del área de la modernización o reingeniería de datos dirigida por modelos. Por tanto, debemos contrastar nuestra contribución con trabajos en los que se aplican técnicas basadas en modelos a la reingeniería de datos, y con los resultados obtenidos antes de aplicar las técnicas MDE, por ejemplo en análisis del código que accede a datos y en la automatización de la normalización de datos. Dado que se trata de un área que ha emergido recientemente, el número de trabajos sobre modernización de datos basado en modelos es muy reducido. Metamodelos y transformaciones de grafos son usados en [16] donde se plantea una arquitectura basada en transformaciones para una migración de Oracle Forms a .NET. Sin embargo, no se usan los estándares de MDE y algunas tareas como la inyección de modelos y generación de código no se implementan como transformaciones. En [17] se describe un enfoque de reingeniería de datos en el que a partir del esquema físico de la base de datos se obtiene un modelo conforme a un metamodelo independiente de la plataforma. Este modelo es transformado en un diagrama de clases que representa un posible esquema conceptual de la base de datos y finalmente este diagrama sirve para crear la nueva aplicación. Esta propuesta es más limitada que la nuestra dado que no contempla aspectos como la detección de errores a partir del análisis de los datos y del código, o la posibilidad de la normalización del esquema obtenido.

DB-MAIN es una herramienta para ingeniería inversa de datos que ofrece una funcionalidad muy completa que incluye extractores del esquema de la base de datos legacy, transformaciones entre esquemas, herramientas de análisis de código y datos, visualizadores de datos. A lo largo de los años, el desarrollo de esta herramienta ha dado lugar a un buen número de contribuciones en el área de reingeniería de datos. Por ejemplo, en [18] se describen estrategias para la reingeniería de un sistema de información legado que abordan tanto la migración de los programas como de los datos. En la conversión del esquema se aplica una estrategia de transformación pero no se aplican técnicas de MDE sino que un parser DDL extrae el esquema físico de la base de datos y entonces se analizan los programas y los datos para extraer la información necesaria para refinar el esquema obtenido en un esquema lógico que finalmente es convertido en el esquema conceptual. Se usa el toolkit de transformaciones de esquema que proporciona DB-MAIN. Sin duda, DB-MAIN es la principal referencia con la que comparar nuestro framework. El grado de automatización e interoperabilidad conseguido con las técnicas MDE es superior al conseguido con DB-MAIN.

Existen algunos trabajos que abordan el problema de automatizar el proceso de normalización de bases de datos, pero no siguen un enfoque MDE. En [19] se propone un método que mediante la definición de un metamodelo UML para esquema de datos, declaraciones OCL para definir las formas normales y declaraciones de reglas para reescribir grafos, consiguen especificar los pasos de la transformación entre formas normales. Sin embargo, no aborda el descubrimiento de las dependencias funcionales que hemos implementado mediante la aplicación de la teoría FCA (Formal Concept Analysis) [14] utilizando la herramienta ConExp [20] y [21].

7 Conclusiones y trabajo futuro

En este artículo se presenta un framework para dar soporte a la ejecución de un proceso de modernización de datos relacionales basado en modelos, donde se soportan varias mejoras de calidad y la migración de bases de datos legacy. La estructura modular de la solución facilita la extensión permitiendo configurar el soporte de las etapas de un proceso de reingeniería (ingeniería inversa, reestructuración y generación del sistema). Se han implementado varios módulos para las tareas de identificación de errores y para la generación de código. El nivel de automatización obtenido es alto y tan solo tras la identificación de errores, de manera opcional, se solicita la asistencia del usuario de la modernización para confirmar las posibles mejoras.

Como trabajo futuro, se pueden aplicar comprobaciones para evitar inconsistencias al excluir registros que no verifiquen la incorporación de las nuevas restricciones tras la etapa de corrección de errores. Otra línea de mejora consiste en considerar las refactorizaciones incluidas en [22]. Por otro lado el registro de la trazabilidad durante el proceso de normalización nos permitirá la generación del script DML. También considerar incluir todos los elementos de la base de datos en el proceso de modernización (vistas, secuencias, procedimientos de base de datos, etc.). Por último, para reducir el tamaño de los modelos de datos se pueden aplicar técnicas de *data mining*.

References

1. ADM, Architecture-Driven Modernization, http://adm.omg.org
2. Cánovas, J.L. and García-Molina, J.; "An Architecture-Driven Modernization Tool for Calculating Metrics", IEEE Software, julio 2009.
3. Seacord, R. et. al., "Modernizing Legacy Systems: Software Technologies, Engineering Process and Business Practices",Addison-Wesley, 2003.
4. Henrard, J. et al., "Strategies for Data Reengineering", WCRE, 2002, 211-220
5. Hainaut, J.L., "Legacy and Future of Data Reverse Engineering", WCRE, 2009, 4.
6. EJB 3 JPA Specification. http://jcp.org/aboutJava/communityprocess/final/jsr220
7. Chikofsky, E. J. and J. H. C. II, "Reverse engineering and design recovery: A taxonomy," IEEE Software, vol. 7, pp. 1317, 1990.
8. Cánovas, J.L., García-Molina, J., "A Domain Specific Language for Extracting Models in Software Modernization". ECMDA-FA 2009: 82-97.
9. Sánchez, J., García-Molina, J. and Menárguez, M., "RubyTL: A Practical, Extensible Transformation Language". 2nd European Conference on Model Driven Architecture, volume 4066, pages 158172. Lecture Notes in Computer Science, June 2006.
10. Henrard, J. et al., "Program understanding in databases reverse engineering", DESA, 1998, Volume 1460/1998, 70-79
11. Elmasri, R., et. al.: "Fundamentals of Database Systems". Addison-Wesley, 2000.
12. The Concept Explorer. http://conexp.sourceforge.net/
13. Formal Concept Analysis. http://www.upriss.org.uk/fca/fca.html
14. Baixeries, J., "A Formal Concept Analysis framework to mine functional dependencies". Workshop on MML , Villa Geno,Como, Italy, 2004.

15. MOFScript Project. http://www.eclipse.org/gmt/mofscript/
16. Andrade, L.F. et. al., "Forms2net - migrating oracle forms to Microsoft .net". GTTSE, pages 261_277, 2006.
17. Polo, M., Rodriguez de Guzmán, I. G., Piattini, M., "An MDA-based approach for database re-engineering". Journal of Software Maintenance (SMR), 2007
18. Hainaut, J.L., Cleve, A., Henrard, J. and Hick, J.M., "Migration of Legacy Information Systems", T. Mens y S. Demeyer (eds.), Software Evolution, 105-138, Springer, 2008.
19. Akehurst, D.H., Bordbar, B., Rodgers, P. J. and G. Dalgliesh, N. T., "Automatic Normalisation via Metamodelling". ASE 2002 Workshop on Declarative Meta Programming to Support Software Development, September 2002.
20. Janosi Rancz, K. T. and Varga, V., "A method for mining functional dependencies in relational database design using FCA". Studia Universitatis Babes-Bolyai Cluj-Napoca, Informatica, vol. LIII, No. 1, (2008) 17-28, ISSN: 1224-869X .
21. Varga, V. and Janosi Rancz, K. T., "A Software Tool to Transform Relational Databases in Order to Mine Functional Dependencies in it Using Formal Concept Analysis". Proceedings of the Sixth International Conference on Concept Lattices and Their Applications, 21-23 oct. 2008, 1-9.
22. Ambler, S.W. and Sadalage, P.J. "Refactoring Databases: Evolutionary Database Design". Addison Wesley. 2009

iTrace: un framework para soportar el análisis de información de trazabilidad en proyectos de Desarrollo Software Dirigidos por Modelos

Iván Santiago, Juan M. Vara, Valeria de Castro, Esperanza Marcos,

Grupo de Investigación Kybele, Departamento de Lenguajes y Sistemas Informáticos II, Universidad Rey Juan Carlos, Avda. Tulipán S/N, 28933, Móstoles, Madrid, España

{ivan.santiago, juanmanuel.vara, valeria.decastro, esperanza.marcos}@urjc.es

Resumen. El papel clave de los modelos en cualquier proceso de Desarrollo de Software Dirigido por Modelos (DSDM) proporciona un nuevo escenario para manejar la trazabilidad. Hasta ahora, existen una serie de propuestas dedicadas al almacenamiento, visualización, generación semi-automática y gestión de operaciones CRUD con enlaces de traza. No obstante, existe una falta de propuestas que se centren en el análisis de la información proporcionada por tales trazas. Además, las propuestas que llevan a cabo algún tipo de análisis de la información de trazabilidad no tienen en cuenta que esta información es consumida por diferentes tipos de actores, cada uno con sus propias necesidades. Para abordar estas cuestiones en este trabajo se introduce iTrace, un framework para la gestión y el análisis de la información de trazabilidad en proyectos de DSDM. Nuestra propuesta busca mejorar el uso que se hace de la información de trazabilidad disponible en proyectos de DSDM mediante dos tipos diferentes de análisis: análisis orientado a modelos, para modeladores, desarrolladores y el resto de perfiles operativos y análisis orientado a datos, para jefes de proyecto, analistas de negocio y usuarios finales en general.

Palabras Claves: Desarrollo Software Dirigido por Modelos, Análisis de Información de Trazabilidad, Análisis Orientado a Modelos, Análisis Orientado a Datos, Modelos de Traza

1 Introducción

La trazabilidad [1] ha sido siempre un tema relevante en la Ingeniería del Software [2]. Mantener enlaces entre los requisitos, los artefactos de análisis y diseño, el código o los casos de uso, ha resultado útil como una forma de llevar a cabo las pruebas de regresión, la validación de requisitos, etc. Del mismo modo, una gestión adecuada de la información de trazabilidad es clave para controlar la evolución de los diferentes componentes del sistema a lo largo del ciclo de vida del desarrollo software [3].

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 491-504, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

Desafortunadamente, el mantenimiento de estos enlaces es un proceso tedioso, lento y propenso a cometer errores si no se proporcionan herramientas que automaticen total o parcialmente la tarea [4]. Como consecuencia, la información de trazabilidad se convierte en obsoleta muy rápidamente durante el desarrollo del software y, a veces, se omite completamente. La llegada de la Ingeniería Dirigida por Modelos (IDM) [5] puede cambiar drásticamente esta situación. El impacto de la IDM se ha traducido en la aparición de una serie de propuestas metodológicas para el Desarrollo Software Dirigido por Modelos (DSDM) [6], donde el rol clave que juegan los modelos puede influir positivamente en la gestión de la información de trazabilidad, ya que en un proceso de DSDM las trazas entre artefactos software que tienen que ser mantenidas son, principalmente, enlaces entre los elementos de los diferentes modelos manejados a lo largo del proceso.

En este sentido, en trabajos previos [7] hemos llevado a cabo una revisión sistemática de la literatura para evaluar el estado del arte sobre la gestión de la trazabilidad en el contexto del DSDM. Una de las principales conclusiones de dicha revisión fue que las operaciones más comúnmente tratadas por las propuestas existentes son el almacenamiento, la visualización y las operaciones de creación, recuperación modificación y borrado (CRUD, *Create*, *Read*, *Update* y *Delete*) de enlaces de traza. Por el contrario, las operaciones menos comúnmente tratadas son el intercambio y el análisis de la información de trazabilidad.

Además, las pocas propuestas que realizan o proponen algún tipo de análisis, no consideran que la información de trazabilidad es "consumida" por diferentes tipos de actores, cada uno con sus propias necesidades y objetivos respecto a la información de trazabilidad [8].

Para hacer frente a este problema, en este trabajo se introduce el primer prototipo de iTrace: un framework para soportar la gestión y el análisis de información de trazabilidad en proyectos de DSDM. Para dar respuesta a las necesidades de los diferentes actores respecto a la información de trazabilidad, iTrace soporta dos tipos de análisis. Por una parte, los modeladores, desarrolladores y el resto de perfiles operativos necesitan de información de bajo nivel. Para hacer frente a sus necesidades, iTrace soporta el Análisis Orientado a Modelos, cuyos resultados se proporcionan en forma de modelos de traza que pueden ser posteriormente procesados mediante técnicas de IDM [9]. Por otra parte, las necesidades de los analistas de negocio, jefes de proyecto y usuarios finales van más en la línea de disponer de datos que apoyen los procesos de toma de decisiones y que permitan elicitar conocimiento a partir de la información de trazabilidad (de bajo nivel) presente en cualquier proyecto de DSDM. Para hacer frente a estas necesidades, iTrace soporta el Análisis Orientado a Datos que produce información de alto nivel en forma de datos agregados.

El resto del trabajo se estructura de la siguiente forma. La sección 2 revisa los trabajos relacionados en el área. La sección 3 presenta la propuesta iTrace, detallando el proceso y los componentes técnicos que la conforman. La sección 4 ilustra la propuesta mediante un caso de estudio, mientras que la sección 5 discute las limitaciones actuales e identifica las principales líneas de trabajo futuro para solventarlas. Finalmente, la sección 6 resume las principales conclusiones derivadas de este trabajo.

2 Trabajos relacionados

Antes de comenzar la revisión de los trabajos relacionados, nos gustaría concretar algunos de los términos relacionados que se utilizan a lo largo de este trabajo. Definimos una *relación de trazabilidad* como una correspondencia entre dos o más tipos de elementos (por ejemplo, entre clases). Por otro lado, nos referimos a las instancias de estas relaciones como *enlaces de traza* (o simplemente trazas). Finalmente, la *información de trazabilidad* es generalmente obtenida a partir de uno o más enlaces de traza; es decir, los enlaces de traza son la materia prima para la construcción de información de trazabilidad.

Existen varios trabajos relacionados con la gestión de la trazabilidad en el contexto del DSDM. La mayoría de estos trabajos consideran el almacenamiento, visualización, generación semi-automática y la realización de operaciones CRUD de enlaces de traza, pero solo unos pocos de esos trabajos abordan el análisis de la información de trazabilidad que proporcionan dichas trazas. Estos trabajos pueden ser clasificados en tres grandes categorías de acuerdo con su objetivo principal: 1) análisis del impacto [10–12]; 2) generación de informes de trazabilidad [13, 14] e 3) identificación y clasificación de trazas [15].

Respecto a los trabajos centrados en el análisis del impacto, los trabajos de Anquetil et al. [10] y Walderhaug et al. [12] presentan propuestas para la extracción de información de un repositorio de enlaces de traza con el objetivo de identificar todos los artefactos potencialmente afectados por un cambio. Además, el primer trabajo acompaña la propuesta de una herramienta de soporte. Por otra parte, Olsen y Oldevik [11] presentan un prototipo donde los artefactos enlazados son mostrados al seleccionar los elementos del modelo origen. Estas propuestas ayudan en la identificación de los elementos que pueden verse afectados por un determinado cambio, ofreciendo el mismo tipo de información independientemente del actor que solicita dicha información.

A continuación consideramos dos herramientas diferentes centradas en la generación de informes de trazabilidad: TraceTool and Safety Requirements Trace Tool. La primera herramienta, presentada por Valderas y Pelechano en [14], genera informes de trazabilidad en formato HTML que describen cómo los elementos de un modelo navegacional se derivan de un modelo de requisitos. La segunda herramienta, presentada en [13] por Sánchez et al., consume modelos de traza para generar informes de trazabilidad en formato HTML, en el contexto del DSDM para robots de servicios tele-operados. Nótese que ambas herramientas están centradas en la trazabilidad de requisitos, es decir, no consideran la trazabilidad de más bajo nivel, como las trazas que pueden ser derivadas de una transformación de modelos de bajo nivel o una transformación modelo a texto.

Finalmente, en [15] Paige et al. presentan *Traceability Elicitation and Analysis Process* (TEAP), una propuesta para la identificación y clasificación de relaciones de trazabilidad. Clasificar los enlaces de trazabilidad ayuda a entenderlos y manejarlos. No obstante, a pesar de tratarse de una propuesta muy útil para los miembros de la comunidad de la IDM, puede ser difícil de comprender para un analista de negocio o un usuario final.

iTrace proporciona una serie de contribuciones respecto al estado del arte actual: en primer lugar, la propuesta se centra en los artefactos software de un proyecto de DSDM que implícitamente definen las relaciones de trazabilidad que deben ser monitorizadas (además de los requisitos, ampliamente abordados en la literatura): no solo transformaciones de modelo a modelo, sino también modelos de anotación y modelos de *weaving* en general [9]. También soporta dos tipos de análisis para dos grupos principales de actores: Análisis Orientado a Modelos y Análisis Orientado a Datos. Además, podríamos decir que todos los trabajos revisados, salvo el de Walderhaug et al. [12] proporcionan un análisis ad-hoc. Es decir, el proceso de análisis propuesto encaja en sus propios proyectos de DSDM, mientras que iTrace pretende proporcionar una propuesta genérica que se pueda aplicar a cualquier proyecto existente de DSDM. Por último, otra de las características relevantes en relación a las propuestas existentes es el hecho de que iTrace soporta el análisis multidimensional de los enlaces de traza.

3 iTrace: un framework para soportar análisis de información de trazabilidad

Esta sección presenta la propuesta para soportar el análisis de información de trazabilidad que se introduce en este trabajo, presentando para ello los principales componentes del entorno iTrace y el proceso de generación y análisis de trazas que soporta.

3.1 Proceso soportado iTrace

El punto de partida del proceso de análisis soportado por iTrace es un proyecto de DSDM existente. Téngase en cuenta que la propuesta pretende ser aplicable a cualquier tipo de proyecto. Por lo tanto, consideramos como punto de partida un proyecto genérico, es decir, un proyecto que incluye modelos a diferentes niveles de abstracción (incluidos modelos de *weaving*) además de un conjunto de transformaciones de modelos que los conectan (transformaciones modelo a modelo y modelo a texto), donde los modelos iniciales (modelos con nivel de abstracción más alto) se transforman posteriormente en modelos de nivel inferior, hasta que su nivel de abstracción permite utilizarlos para generar código.

La parte izquierda de la Fig. 1, muestra una versión simplificada de un proyecto de este tipo, compuesto por tres modelos llamados *Ma*, *Mb* y *Mc* y dos transformaciones de modelo a modelo (*Ma2Mb* y *Mb2Mc*).

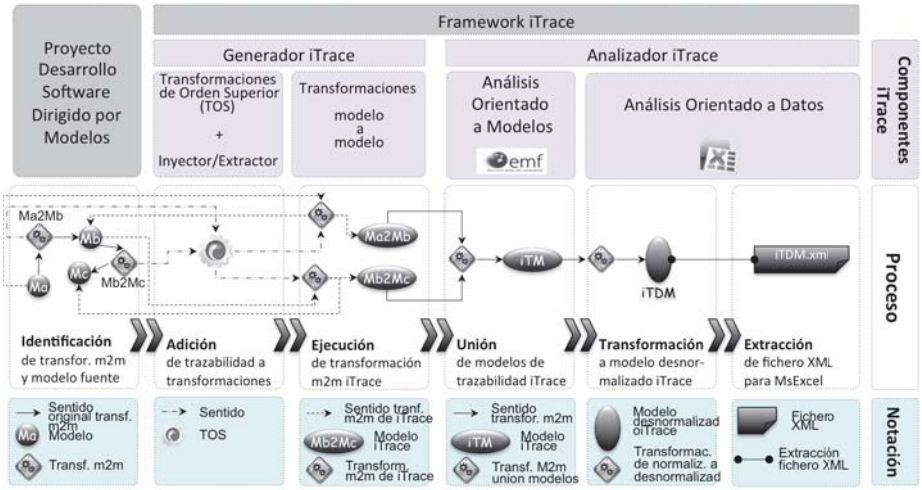


Fig. 1. Visión general del entorno iTrace[1]

El resto de la Fig. 1 proporciona una visión tabular del framework iTrace: la fila superior muestra los **Componentes** involucrados en cada paso del proceso; la siguiente fila muestra los artefactos (modelos, transformaciones, etc.) usados y producidos en cada paso del **Proceso**; finalmente, la última fila muestra la **Notación** utilizada para comprender los elementos gráficos que se utilizan en cada celda.

Si nos centramos en el proceso, la fase de **Identificación** representa el inicio del mismo, donde un proyecto existente de DSDM compuesto por modelos y transformaciones es seleccionado como entrada. Durante la fase de **Adición**, cada transformación del proyecto es enriquecida mediante una Transformación de Orden Superior (TOS) [16] siguiendo la idea esbozada por Jouault en [17]. A continuación, las transformaciones enriquecidas son ejecutadas durante la fase de **Ejecución** con el fin de generar modelos de traza además de los correspondientes modelos destino. Estos modelos de traza son consolidados en un único modelo durante la fase de **Unión**. Este modelo agregado de trazas es el que sirve de entrada para llevar a cabo el Análisis Orientado a Modelos soportado por iTrace.

Podemos decir que este modelo agregado de trazas es un modelo normalizado. Sin embargo, a la hora de ejecutar las consultas que realizan el Análisis Orientado a Datos soportado por iTrace hemos encontrado más conveniente utilizar un modelo de trazas desnormalizado. Adoptamos esta idea del área de las bases de datos, donde los modelos de datos normalizados son más convenientes para adiciones, modificaciones y borrados, mientras que se utilizan modelos de datos desnormalizados para mejorar el rendimiento de las consultas en sistemas de consulta intensiva, como los almacenes de datos (datawarehouses) [18]. En cierto sentido, el análisis orientado a datos de información de trazabilidad puede ser visto como el análisis de los datos históricos recopilados durante el desarrollo del proyecto, principalmente en forma de modelos de traza. Por lo tanto, durante la fase de **Transformación**, una transformación modelo a

[1] Todas las imágenes se encuentran disponibles a tamaño completo para su mejor visualización en: http://www.kybele.etsii.urjc.es/JISBD/iTrace/

modelo consume el modelo de trazas normalizado generado en la fase de unión (modelo normalizado iTrace) para producir un modelo de trazas desnormalizado (modelo desnormalizado iTrace).

Por último, durante la fase de **Extracción**, el modelo desnormalizado de trazas es serializado en un fichero XML que después será importado a una hoja de cálculo de Excel con el fin de poblar la tabla dinámica que proporciona la base sobre la que se realiza el Análisis Orientado a Datos en la versión actual del prototipo. Nótese que el objetivo de este trabajo era fundamentalmente evaluar la viabilidad de nuestra propuesta. En trabajos futuros evaluaremos la posibilidad de utilizar sistemas más sofisticados para soportar el análisis orientado a datos.

Conviene destacar que hasta ahora se ha presentado un proceso independiente de la tecnología para el análisis de trazabilidad, es decir, el proceso descrito podría ser implementado sobre cualquier marco de metamodelado que ofrezca soporte para el desarrollo de transformaciones de modelos (tanto de modelo a modelo, como de modelo a texto).

A continuación, profundizamos en los detalles de la propuesta mostrando, a modo de prueba de concepto, la implementación de los diferentes artefactos software que componen iTrace utilizando Eclipse Modeling Framework (EMF) [19] como base tecnológica.

3.2 Modelos iTrace

A lo largo del proceso soportado por iTrace se generan dos tipos de modelos de traza: modelos de traza normalizados y modelos de traza desnormalizados. A continuación se presentan brevemente los metamodelos correspondientes para ilustrar la sintaxis abstracta de estos modelos. En cuanto a la sintaxis concreta, hasta ahora utilizamos una versión mejorada de los editores en forma de árbol generados por EMF.

Metamodelo iTrace

El metamodelo de iTrace, que se muestra en la Fig. 2, soporta el modelado de la información de trazabilidad de bajo nivel obtenida a partir de transformaciones de modelos (modelo a modelo y modelo a texto) y modelos *weaving*.

Un `iTraceModel` (clase raíz) contiene `Artifacts` (software) y/o `TraceLinks`. El primero representa los bloques de construcción del proyecto de DSDM, es decir, modelos y código fuente, mientras que el segundo representa las trazas entre ellos. En realidad, estas trazas conectan sus componentes, representados mediante objetos `TraceLinkElement` en el caso de los modelos y `Blocks` en el caso del código fuente. Cada objeto `TraceLinkElement` tiene una referencia `EObject` para apuntar a un elemento del modelo en particular.

Por su parte, cada traza puede ser un `TransformationLink`, derivada de una transformación de modelos, o un `AnnotationLink`, derivada de un modelo de *weaving*. A su vez, la clase `TransformationLink` se especializa en las clases `M2MLink` y `M2TLink`. Mientras que un objeto `M2MLink` relaciona dos o más elementos del modelo (al menos un `TraceLinkSourceElement` y un `TraceLinkTargetElement`), un objeto `M2TLink` relaciona uno o más elementos del modelo con algún bloque de código fuente (`M2TLink.codeTarget.blockCode`).

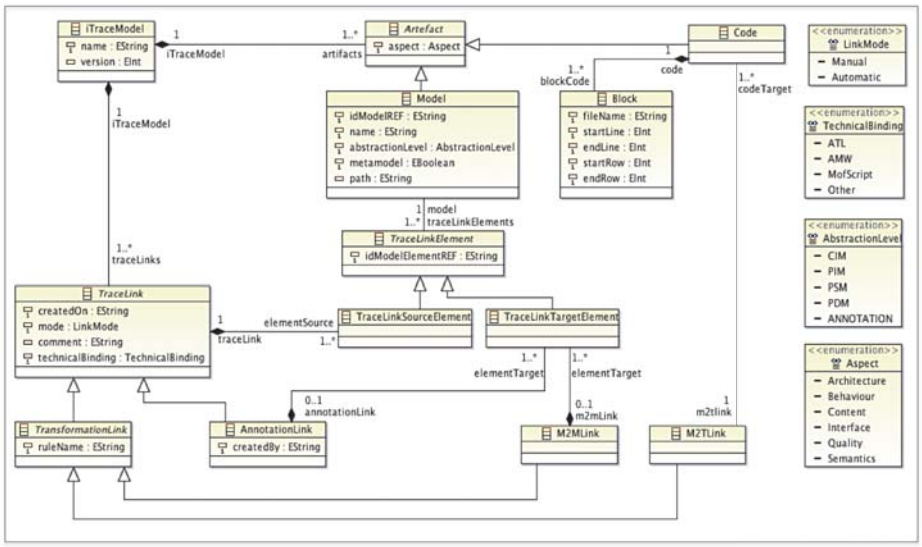


Fig. 2. Metamodelo iTrace

Metamodelo desnormalizado iTrace

El metamodelo desnormalizado de iTrace está diseñado para soportar un paso intermedio en el proceso soportado por iTrace. En particular, la información de trazabilidad recogida en los diferentes modelos iTrace (en la siguiente sección se mostrará como se generan) son consumidos por una transformación ATL que produce información "agregada" en forma de un modelo desnormalizado.

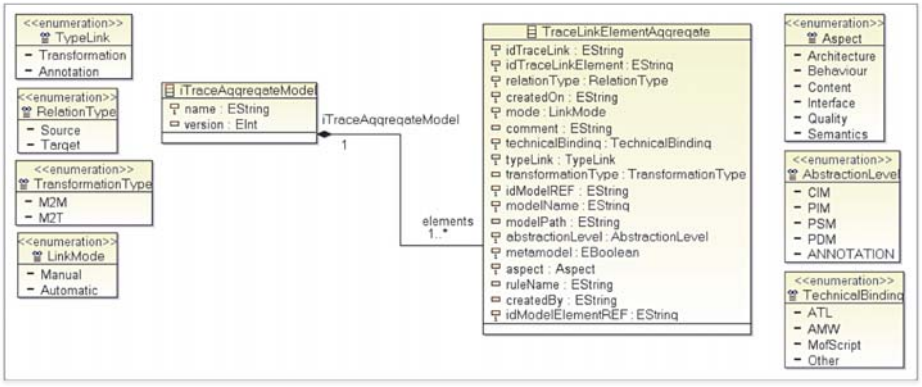


Fig. 3. Metamodelo desnormalizado iTrace

El metamodelo correspondiente, que se muestra en la Fig. 3, contiene dos clases, además de siete tipos de datos enumerados. De esta manera, cada `iTraceDenormalizedModel` se compone de objetos `DenormalizedTraceLinkElements` que poseen un conjunto de atributos que toman su valor de los diferentes tipos enumerados, facilitando las consultas que soportan el Análisis Orientado a Datos.

3.3 Generación de trazas en iTrace

Como se muestra en la Fig. 1, podemos distinguir dos componentes principales en la arquitectura de iTrace: el Generador iTrace diseñado para soportar la extracción de enlaces de traza de un proyecto existente de DSDM, y el Analizador iTrace que procesa la información proporcionada por tales trazas. A continuación presentamos el primer componente basado en el uso de Transformaciones de Orden Superior.

Tenga en cuenta que la versión actual de iTrace soporta la extracción de enlaces de traza a partir de transformaciones modelo a modelo (m2m) escritas en ATL [20], transformaciones modelo a texto (m2t) escritas en lenguaje MOFScript [21] y modelos de *weaving* elaborados con la herramienta Atlas Model Weaver (AMW) [22]. Sin embargo, vale la pena señalar que extender la propuesta para soportar cualquier otro lenguaje de transformación de modelos es técnicamente factible. De hecho, sería casi inmediato si se tratase de un lenguaje basado en un metamodelo de forma que soportase el modelado de las transformaciones incluidas en el proyecto de DSDM en cuestión.

iTrace se apoya en una Transformación de Orden Superior (TOS) [16] para enriquecer las transformaciones m2m existentes para que sean capaces de producir, no solo los modelos destino correspondientes, sino también modelos de traza. Esta idea fue propuesta por Jouault en [17], donde presentaba un prototipo inicial que permitía añadir mecanismos para la generación de trazas a transformaciones ATL existentes. En este caso, el proceso de enriquecimiento de transformaciones m2m soportado por iTrace es un poco más complejo que el que se presentaba en [17], debido al incremento de la complejidad de los metamodelos de iTrace.

La Fig. 4 muestra gráficamente el proceso: en primer lugar, el inyector/extractor TCS [23] para ficheros ATL incluido en la plataforma Atlas Model Management Architecture (AMMA) [24, 25] produce un modelo de transformación a partir del código ATL de una determinada transformación (a); este modelo de transformación es enriquecido mediante una TOS con construcciones para la generación de trazas (b) y finalmente el modelo de transformación resultante es serializado en otra transformación ATL (c). Como se mencionó anteriormente, la ejecución de esta transformación enriquecida producirá, no solo los correspondiente modelos de destino, sino también uno o varios modelos de trazas.

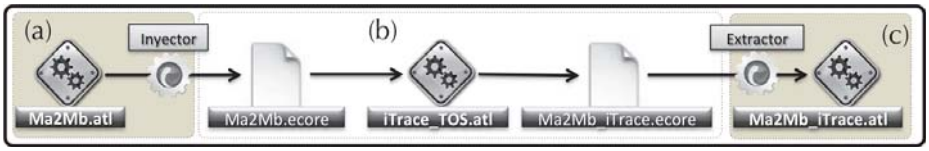


Fig. 4. Adición de capacidades de trazabilidad en transformaciones ATL – adaptación de [17]

3.4 Análisis de trazas en iTrace

Después de mostrar el proceso de generación de enlaces de traza soportado por iTrace, introducimos el proceso de análisis de dichas trazas. Para ello, en esta sección se describen los dos tipos de análisis soportados por iTrace: Análisis Orientado a Modelos y Análisis Orientado a Datos.

Análisis Orientado a Modelos

El Análisis Orientado a Modelos tiene por objetivo ofrecer información que satisfaga las necesidades de los actores involucrados en un proyecto de DSDM con un perfil operacional. Hablamos en general de modeladores, desarrolladores de transformaciones, etc. Los artefactos que suelen manejar este tipo de actores son modelos, por lo tanto el resultado de este tipo de análisis habrá de tomar la forma de modelos. Por ello, la idea subyacente es filtrar los elementos de los modelos de traza existentes atendiendo a ciertos criterios, con el fin de producir nuevos modelos de traza.

Actualmente, existen diferentes formas de consultar el contenido de uno o varios modelos. Las alternativas van desde el uso de GPLs como Java, a aproximaciones más específicas como el uso del lenguaje OCL [26] o herramientas ad-hoc, como EMF Query [27], EMF Query 2 [28] o VirtualEMF [29]. Estos últimos encapsulan gran parte de la complejidad asociada con la navegación de un modelo y podríamos decir que proporcionan las ventajas tradicionales de seguir una aproximación específica de domino en lugar de una aproximación de propósito general [30].

En particular, la implementación actual de iTrace utiliza EMFQuery porque era la iniciativa más madura entre las herramientas evaluadas. Sin embargo, presenta algunas limitaciones en cuanto a la navegación de modelos, ya que solo permite la consulta de objetos directamente anidados en el objeto raíz del modelo (metaclase `iTraceModel` en la Fig. 2). Por lo tanto, la versión actual de iTrace soporta únicamente consultas de recuperación de enlaces de traza, que pueden filtrase de acuerdo a:

- Artefacto software del que se derivan: transformación ATL, transformación MOFScript, modelo AMW, Others.
- Modo de generación: automático, manual.
- Cardinalidad del enlace: 1:1, 1:N, N:1, N:M.

Análisis Orientado a Datos

El objetivo del Análisis Orientado a Datos es obtener información que ofrezca soporte a la toma de decisiones, creando conocimiento a partir de las relaciones de trazabilidad existentes en el proyecto de DSDM bajo análisis. Frente a los modelos de traza (filtrados) producidos por el Análisis Orientado a Modelos, los resultados del Análisis Orientado a Datos son datos textuales.

En general, las organizaciones de desarrollo software tratan de identificar nuevas oportunidades y de mejorar su productividad con el fin de tener una ventaja competitiva. Para ello, es clave disponer de información analítica y estratégica, que se obtiene a partir de los datos que a diario genera y almacena la organización [31]. Este proceso de extracción y análisis de información se lleva a cabo mediante sistemas de inteligencia de negocio [32] que operan sobre soluciones basadas en almacenes de datos [33]. La idea subyacente es utilizar datos operacionales para producir información relevante a nivel táctico o incluso estratégico. En nuestra opinión, sería interesante trasladar este enfoque al uso de la información de trazabilidad que se puede generar en un proyecto de DSDM.

Para ilustrar esta idea, la siguiente sección introduce mínimamente un caso de estudio que describe un escenario básico de Análisis Orientado a Datos de información de trazabilidad.

4 Caso de Estudio

En esta sección se muestra la aplicación de la propuesta en un proyecto de DSDM existente (nótese que por motivos de espacio nos limitamos a mostrar una pequeña parte del caso de estudio).

Para ello, nos apoyamos en M2DAT-DB [34], un framework para Desarrollo Dirigido por Modelos de esquemas de bases de datos modernos que soportan la totalidad del ciclo de desarrollo, desde el nivel PIM hasta el código. En particular, M2DAT-DB incluye soporte para la generación de esquemas ORDB para Oracle y el estándar SQL:2003, así como esquemas XML a partir del modelo conceptual de datos representado con un diagrama de clases UML. En este trabajo nos centramos en tres transformaciones de modelos diferentes: la transformación UML2SQL2003 produce un modelo estándar ORDB (Modelo Específico de Plataforma, PSM) a partir de un diagrama de clases UML (Modelo Independiente de Plataforma, PIM), mientras que la transformación SQL20032ORDB genera un modelo ORDB para Oracle (Modelo Dependiente de Plataforma, PDM) a partir del modelo para el estándar. Por último, la transformación ORDB2SQL2003 implementa la transformación inversa. Además, aparte de los correspondientes modelos origen, cada transformación es capaz de consumir un modelo de anotación para introducir algunas decisiones de diseño en el proceso de transformación [35].

En el proyecto que se analiza con iTrace, estas transformaciones son ejecutadas utilizando el caso de estudio Online Movie Database (OMDB) de Feuerlicht et al. [36]. El uso de un caso de estudio "externo" evita la utilización de modelos ad-hoc que pudieran encajar mejor con nuestros propósitos.

Siguiendo el proceso descrito en la Sección 3.1 se genera un modelo agregado de trazas de trazas que es transformado en un modelo desnormalizado que es la entrada del proceso de Análisis Orientado a Datos. Dicho modelo se serializa en un fichero XML que se utiliza para poblar la tabla dinámica de MS Excel que soporta el Análisis Orientado a Datos.

La Fig. 5 muestra la información que permite analizar el rol y el nivel de abstracción de los modelos involucrados en las diferentes transformaciones. La parte izquierda de la figura muestra la información de forma tabular (1):

- En la parte superior se muestran los criterios de alto nivel que pueden ser utilizados. De esta forma, la información generada puede ser filtrada de acuerdo a la versión del modelo de trazas (1.0 en este caso); la fecha de creación (20 de Diciembre de 2011); el modo de creación (automático); o el aspecto modelado por los modelos considerados (contenido).
- El eje vertical corresponde al nivel de abstracción de los elementos del modelo trazado (PDM, PIM o PSM) y pueden ser refinados de acuerdo al rol de los elementos del modelo en el enlace de traza (origen o destino).
- El eje horizontal considera las diferentes transformaciones involucradas en el proyecto. Se puede ofrecer mayor nivel de detalle incluyendo los modelos concretos implicados en dichas transformaciones.

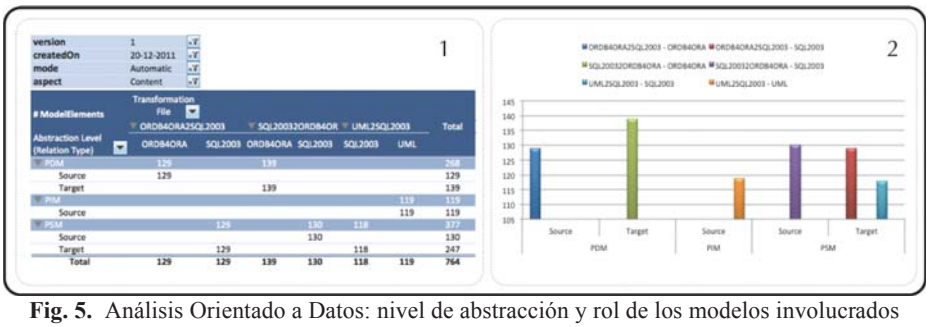


Fig. 5. Análisis Orientado a Datos: nivel de abstracción y rol de los modelos involucrados

Por ejemplo, la figura muestra que existen 129 enlaces de traza generados por la transformación ORDB4ORA2SQL2003 que apuntan a elementos del modelo ORDB4ORA con nivel de abstracción PDM y que son el origen de un enlace de traza.

Finalmente, la parte derecha de la figura (2) proporciona una representación gráfica agregada de los mismos datos. Esta representación es más conveniente para obtener una rápida visión general de los valores más destacados. Por ejemplo, observamos que el número objetos del modelo ORDB4ORA que son referenciados por trazas generadas por la transformación SQL20032ORDB4ORA destaca sobre el resto (barra vertical verde en la figura).

5 Limitaciones y trabajos futuros

En este trabajo hemos presentado un primer prototipo de un entorno para soportar el análisis de información de trazabilidad en proyectos de DSDM. Una vez comprobada la viabilidad de la propuesta, pasamos a evaluar las principales limitaciones del prototipo actual y la forma de abordar las futuras mejoras.

Aunque iTrace esta diseñado para obtener información de trazabilidad a partir de transformaciones modelo a modelo y modelo a texto, la versión actual solamente trabaja con modelos de *weaving* y transformaciones modelo a modelo. Por lo tanto, el siguiente paso es soportar la generación automática de enlaces de traza a partir de transformaciones modelo a texto. En principio, esta funcionalidad no debería entrañar una excesiva complejidad ya que existen trabajos, como de Oldevik y Neple [21] que ya la soportaban, aunque aún no hemos evaluado sus resultados.

Otro objetivo perseguido con iTrace es ofrecer soporte para consultar diferentes versiones de un mismo modelo de trazas. Disponer de diferentes versiones de un mismo modelo de traza permitiría controlar la evolución de la información de trazabilidad a lo largo del proyecto, simplificando algunas tareas como por ejemplo el análisis de huérfanos.

En cuanto al Análisis Orientado a Modelos, en la sección 3.4 se mencionaba que, de entre las diferentes formas de consultar modelos (de traza) que existían, se ha optado por utilizar EMF Query debido a que era la propuesta más madura para la consulta de modelos EMF. Sin embargo, se señalaban algunas limitaciones respecto a sus capacidades de navegación que, de alguna manera, limitan el tipo de consultas que pueden ser ejecutadas. Por lo tanto, estamos evaluando otros motores de consulta

como EMF Query2 y el uso de lenguajes imperativos para gestión de modelos como Epsilon Object Language (EOL) [37].

Respecto al Análisis Orientado a Datos, aunque actualmente utilizamos tablas dinámicas de MS Excel para soportarlo, la idea es construir el componente de Análisis Orientado a Datos sobre un almacén de datos para incrementar la potencia y alcance del análisis a la manera de los sistemas de inteligencia de negocio.

Por último, la visualización actual de los modelos iTrace (tanto el normalizado como el desnormalizado) se basa en la utilización del editor básico en forma de árbol proporcionado por EMF. Sin embargo, creemos que una representación multi-panel similar a la ofrecida por AMW [22] se ajusta mejor a la naturaleza "relacional" de los modelos de traza, cuya finalidad es representar relaciones entre los elementos de dos o más modelos. Para ello, ya hemos desarrollado un primer prototipo de un editor multi-panel que soporta la creación de enlaces de traza mediante *drag & drop* de los elementos de los modelos trazados.

6 Conclusiones

La llegada de la IDM [5] proporciona un nuevo escenario que puede ayudar a hacer realidad las ventajas que la adecuada gestión de la trazabilidad aportaría al desarrollo de software [2]. El hecho de que los modelos sean los principales artefactos manipulados a lo largo del proceso de desarrollo y de que estén conectados mediante transformaciones de modelos permite derivar semiautomáticamente las trazas entre los diferentes artefactos [5].

Sin embargo, el estudio de la literatura existente pone de manifiesto la falta de propuestas centradas en un uso adecuado de la información de trazabilidad que puede ser recogida de cualquier proyecto de DSDM. Además, las pocas propuestas que abordan el tema no consideran los diferentes tipos de actores involucrados en el proceso de desarrollo. Por ejemplo, las necesidades de un desarrollador de transformaciones de modelos pueden ser diferentes a las necesidades de un jefe de proyecto.

Para abordar estos problemas, este trabajo ha presentado iTrace, un framework para soportar el análisis de información de trazabilidad automáticamente generada a partir de un proyecto de DSDM. Más concretamente, iTrace soporta dos tipos de análisis, llamados Análisis Orientado a Modelos y Análisis Orientado a Datos.

El primero está orientado a aquellos actores que poseen un perfil más operacional y su principal característica es que los resultados se proporcionan en forma de modelos que, a su vez, pueden ser procesados utilizando técnicas de IDM, tales como las transformaciones de modelos [9] para filtrar la información que contienen.

Por otra parte, el objetivo del Análisis Orientado a Datos es agregar información de bajo nivel para proporcionar datos que pueden ser usados para dar soporte al proceso de toma de decisiones. Por ejemplo, un jefe de proyecto puede usar el análisis mostrado en el caso de estudio para identificar si el número trazas que apunta a los objetos de un determinado modelo es excepcionalmente elevado. Si así fuera, podemos inferir que el impacto de cualquier modificación realizada sobre dicho modelo afectará a muchos otros artefactos del proyecto. En tal caso, pudiera convenir descomponer

dicho modelo en modelos más pequeños que redujesen el impacto de cualquier modificación.

Finalmente, nos gustaría hacer hincapié en el hecho de que iTrace es solamente un prototipo preliminar. A pesar de ser completamente funcional, todavía deja mucho margen de mejora. En este sentido, este trabajo puede ser visto como un *position paper* que ha servido para comprobar la viabilidad de la propuesta y para identificar las principales áreas de trabajo futuro. Consideramos que los resultados son prometedores y ya estamos trabajando en el desarrollo de las diferentes mejoras.

Agradecimientos

Este trabajo se ha llevado a cabo en el marco del proyecto MASAI (Ref. TIN-2011-22617) financiado por el Ministerio de Ciencia y Tecnología de España.

Referencias

1. IEEE Standard Glossary of Software Engineering Terminology. IEEE Std 610.12-1990. 1 (1990).
2. Ramesh, B., Stubbs, C., Powers, T., Edwards, M.: Requirements traceability: Theory and practice. Annals of Software Engineering. 3, 397-415 (1997).
3. Asuncion, H., Asuncion, A., Taylor, R.: Software traceability with topic modeling, (2010).
4. Hayes, J.H., Dekhtyar, A., Sundaram, S.K.: Advancing candidate link generation for requirements tracing: the study of methods. Software Engineering, IEEE Transactions on. 32, 4-19 (2006).
5. Bezivin, J.: In search of a basic principle for model driven engineering. Novatica Journal, Special Issue. (2004).
6. Stahl, T., Voelter, M., Czarnecki, K.: Model-Driven Software Development: Technology, Engineering, Management. John Wiley & Sons (2006).
7. Santiago, I., Jiménez, Á., Vara, J.M., Castro, V.D., Bollati, V.A., Marcos, E.: Model-Driven Engineering As a New Landscape For Traceability Management: A Systematic Review. Information and Software Technology (Submitted). (2012).
8. Aizenbud-Reshef, N., Nolan, B.T., Rubin, J., Shaham-Gafni, Y.: Model traceability. IBM Systems Journal. 45, 515-526 (2006).
9. Bernstein, P.: Applying model management to classical meta data problems. First Biennial Conference on Innovative Data Systems Research. , Asilomar, CA, USA (2003).
10. Anquetil, N., Kulesza, U., Mitschke, R., Moreira, A., Royer, J.C., Rummler, A., Sousa, A.: A model-driven traceability framework for software product lines. Software and Systems Modeling. 9, 427-451 (2010).
11. Olsen, G.K., Oldevik, J.: Scenarios of traceability in model to text transformations, (2007).
12. Walderhaug, S., Johansen, U., Stav, E., Aagedal, J.: Towards a generic solution for traceability in MDD. 2th European Conference on Model Driven Architecture - Traceability Workshop (ECMDA-TW'06). (2006).
13. Sánchez, P., Alonso, D., Rosique, F., Álvarez, B., Pastor, A.: Introducing Safety Requirements Traceability Support in Model-Driven Development of Robotic Applications. IEEE Transactions on Computers. 60, 1059-1071 (2011).
14. Valderas, P., Pelechano, V.: Introducing requirements traceability support in model-driven development of web applications. Information and Software Technology. 51, 749-768 (2009).
15. Paige, R., Olsen, G.K., Kolovos, D., Zschaler, S., Power, C.: Building model-driven engineering traceability classifications. Proceedings of the 4th European Conference on Model Driven Architecture - Traceability Workshop (ECMDA-TW'08). pp. 49-58. , Berlin, Germany (2008).

16. Tisi, M., Cabot, J., Jouault, F.: Improving higher-order transformations support in ATL. International Conference on Model Transformation (ICMT) (2010).
17. Jouault, F.: Loosely coupled traceability for ATL, (2005).
18. Sanders, G., Shin, S.: Denormalization Effects on Performance of RDBMS. Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 3 - Volume 3. p. 3013--. IEEE Computer Society, Washington, DC, USA (2001).
19. Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: EMF: Eclipse Modeling Framework. Addison-Wesley Professional (2008).
20. Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I.: ATL: A model transformation tool. Science of Computer Programming. 72, 31-39 (2008).
21. Oldevik, J., Neple, T., Grønmo, R., Aagedal, J., Berre, A.-J.: Toward Standardised Model to Text Transformations. In: Hartman, A. and Kreische, D. (eds.) Model Driven Architecture – Foundations and Applications. pp. 239-253. Springer Berlin / Heidelberg (2005).
22. Didonet del Fabro, M., Bézivin, J., Valduriez, P.: Weaving Models with the Eclipse AMW plugin. Eclipse Modeling Symposium. (2006).
23. Jouault, F., Bézivin, J., Kurtev, I.: TCS: a DSL for the specification of textual concrete syntaxes in model engineering. Proceedings of the 5th international conference on Generative programming and component engineering. pp. 249-254. ACM, New York, NY, USA (2006).
24. The AMMA Platform, http://www.sciences.univ-nantes.fr/lina/atl/AMMAROOT/.
25. Bézivin, J., Jouault, F., Rosenthal, P., Valduriez, P.: Modeling in the Large and Modeling in the Small. In: Aßmann, U., Aksit, M., and Rensink, A. (eds.) Model Driven Architecture. p. 901. Springer Berlin / Heidelberg (2005).
26. The Eclipse Project: MDT OCL, http://www.eclipse.org/modeling/mdt/?project=.
27. The Eclipse Project: EMF Model Query, http://www.eclipse.org/modeling/emf/?project=query.
28. The Eclipse Project: EMF Model Query 2, http://www.eclipse.org/modeling/emf/?project=query2.
29. Clasen, C., Jouault, F., Cabot, J.: VirtualEMF: A Model Virtualization Tool. In: De Troyer, O., Bauzer Medeiros, C., Billen, R., Hallot, P., Simitsis, A., and Van Mingroot, H. (eds.) Advances in Conceptual Modeling. Recent Developments and New Directions. pp. 332-335. Springer Berlin / Heidelberg (2011).
30. Mernik, M., Heering, J., Sloane, A.M.: When and how to develop domain-specific languages. ACM Computing Surveys (CSUR). 37, 316-344 (2005).
31. Watson, H.J., Goodhue, D.L., Wixom, B.H.: The benefits of data warehousing: why some organizations realize exceptional payoffs. Information & Management. 39, 491-502 (2002).
32. Kimball, R.: The Data Warehouse Lifecycle Toolkit. Wiley (1998).
33. Inmon, W.H.: Building the Data Warehouse. Wiley (1996).
34. Vara, J., Vela, B., Bollati, V., Marcos, E.: Supporting Model–Driven Development of Object–Relational Database Schemas: A Case Study. In: Paige, R. (ed.) Theory and Practice of Model Transformations. pp. 181-196. Springer Berlin / Heidelberg (2009).
35. Vara, J.M., Castro, M.V.D., Fabro, M.D.D., Marcos, E.: Using Weaving Models to automate Model-Driven Web Engineering proposals. International Journal of Computer Applications in Technology. 39, 245-252 (2010).
36. Feuerlicht, G., Pokorný, J., Richta, K.: Object-Relational Database Design: Can Your Application Benefit from SQL:2003? In: Barry, C., Lang, M., Wojtkowski, W., Conboy, K., and Wojtkowski, G. (eds.) Information Systems Development. pp. 975-987. Springer (2009).
37. Kolovos, D., Paige, R., Polack, F.: The Epsilon Object Language (EOL). In: Rensink, A. and Warmer, J. (eds.) Model Driven Architecture – Foundations and Applications. pp. 128-142. Springer Berlin / Heidelberg (2006).

Diseño de Niveles y uso de Motores en el Desarrollo de Videojuegos dirigido por Modelos

Víctor M. Bolinches y José A. Carsí

Grupo de Ingeniería del Software y Sistemas de Información,
Departamento de Sistema Informáticos y de Computación,
Universitat Politècnica de València.
Camí de Vera s/n 46022 Valencia, España

vicboma@ei.upv.es
pcarsi@dsic.upv.es

Resumen: La propuesta del desarrollo de juegos dirigidos por modelos (MDGD) ofrece un multi-modelo para la especificación de videojuegos dividido en varias vistas: jugabilidad, interfaz gráfica y control entre otras. Este concepto de modelado conceptual permite a los diseñadores de juegos especificar juegos a un gran nivel de abstracción independientemente de la plataforma utilizada mediante la aplicación de MDD. En este trabajo se presenta una nueva vista añadida al modelado del *gameplay* de los videojuegos que permite el diseño de niveles. Ya que es común en el desarrollo de juegos el utilizar motores 'estándares' en su construcción, se muestra cómo es posible integrar un motor de *tiles* 2D en el proceso de desarrollo definiendo un meta-modelo específico de la plataforma (PSM) genérico que permite definir la estructura y el comportamiento del sistema haciendo uso del motor sin entrar en los detalles técnicos de implementación.

Keywords: desarrollo de software dirigido por modelos, MDA , diseño de niveles, PIM,PSM, motor de videojuegos 2D, MDD, especificación gameplay, diseño de juegos.

1 Introducción

Hace décadas que la industria del videojuego desarrolla software para todos los públicos en una amplia variedad de plataformas tecnológicas. Lejos quedan las sofisticadas recreativas y videoconsolas de los 80 que ejecutaban multitud de juegos en sus *hardwares*. A sus espaldas tenían grandes equipos de desarrollo que más tarde en los 90 catalogarían a estos videojuegos como juegos AAA por ser sus gráficos, jugabilidad (serie de decisiones interesantes [15].) y sonido de excelencia. A día de hoy, nuevas tecnologías tales como los *Tablets* y los dispositivos móviles de nueva generación son

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 505-518, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

un foco creciente en el desarrollo de videojuegos que hace que un grupo reducido de personas expertas en el dominio de una plataforma pueda realizar juegos en poco tiempo olvidando las largas etapas que conlleva el desarrollo de grandes juegos y de los motores que acompañan a éstos en tiempo de ejecución. Uno de los problemas en el desarrollo de videojuegos viene dado porque se carece de un lenguaje de especificación para los videojuegos con el que los diseñadores que no saben programar, puedan escribir la documentación del diseño [10]. Los programadores traducen como pueden estas especificaciones a código, que más tarde será compilado, ejecutado y apto para la detección de errores.

Con la intención de solventar estos problemas, se propuso en [2] elevar el nivel de abstracción del desarrollo de videojuegos mediante un modelo independiente de la plataforma (PIM) para el diseño de videojuegos. *Model Driven Architecture* (MDA) permite elevar el nivel de abstracción tecnológico puesto que los modelos pueden usarse para favorecer el uso de lenguajes específicos de domino que los diseñadores utilizarán para modelar los artefactos en sus propios conceptos de dominio [6]. Puesto que MDA favorece el proceso de desarrollo de software multi-paradigma y es una arquitectura que proporciona un conjunto de guías para estructurar especificaciones expresadas con modelos, se utiliza un modelo PIM de alto nivel de abstracción a través de lenguajes específicos de dominio (DSLs) para el uso del dominio de videojuegos. Las especificaciones de este modelo PIM son usadas para tener en cuenta detalles de implementación precisando los requerimientos funcionales que lo hace sobrevivir a los cambios que se produzcan en las tecnologías dependientes y arquitecturas software. En este artículo se presenta una nueva especificación añadida al modelo de interactividad de videojuegos como es el diseño de niveles que junto con las tres perspectivas fundamentales: jugabilidad, control e interfaz gráfica de usuario, definen el *gameplay* del juego.

El término "motor de videojuegos" salió a la luz a principios de los 90 cuando grupos de desarrolladores perfeccionaban sus juegos antes de sacarlos al mercado. Estos se dieron cuenta que separar las partes principales del software les permitía especializarse y crecer en los conceptos básicos de un videojuego. El desarrollo de éste ya no se haría desde una etapa principal partiendo desde cero sino que se reutilizaría este desarrollo para lanzar nuevas secuelas de juegos más rápidos y más fáciles de desarrollar. Gracias a los motores pudiendo ser más competentes en la industria del videojuego.

Se presenta también un motor de *tiles* 2D mediante un meta-modelo PSM para conceptualizar la plataforma destino que especifica la estructura y el comportamiento del sistema (juego) sin entrar en detalles técnicos de implementación para el manejo de objetos en 2 dimensiones inmersos en un ambiente y que interactúan entre sí [7].
Trataremos únicamente el desarrollo de videojuegos 2D puesto que el mercado está en auge, sobre todo en los dispositivos móviles y los *Tablets* que a día de hoy son un punto clave para el desarrollo de videojuegos 2D, donde pequeños equipos de desarrollo proponen juegos casuales o de tipo *indies* en breves periodos de tiempo con sus secuelas o sagas correspondientes. La estructura de este artículo es la siguiente: la sección 2 ofrece una visión general del estado del arte en los avances de las metodologías de desarrollo de juegos. La sección 2.1 discute cómo aplicar MDA al desarrollo

de videojuegos, donde se ahonda en la especificación de la interfaz gráfica de usuario (GUI) y se muestran las perspectivas de las entidades y reglas. En la sección 3 se describe el diseño de niveles. En la sección 4 de describe el desarrollo específico de un motor de *tile*s 2D genérico mediante transformaciones PIM a PSM. La sección 5 ofrece las conclusiones de la investigación y los trabajos futuros.

2 Estado del Arte

En esta sección, se discute sobre algunos trabajos existentes relacionados con nuestro propósito:

Furtador et al [1] abordan el desarrollo de videojuegos mediante un DSL y líneas de productos software, todo el código que generan se apoya en un motor llamado *ArcadEngine* que extiende e implementa dicho código proveniente de los modelos de *ArcadEx* que quita complejidad al código generado. Después un segundo motor llamado *FlatRedBall* es el que consume este código final y lo presenta en el framework destino. Los autores argumentan que no generan el juego en su totalidad sino que pretenden usar un DSL y líneas de productos software para apoyarlas en el motor y obtener una reutilización de software más estructurada, eficaz e intuitiva.

El uso de este DSL permite a los diseñadores de juegos elevar el nivel de abstracción y trabajar con conceptos más próximos a su dominio de aplicación, el problema viene dado en que la utilización de un DSL y un motor a bajo nivel les limita la productividad de géneros de videojuegos en una plataforma destino, evitando así el posible uso industrializado donde se desarrollan videojuegos para muchas plataformas.

Hernandez F. et al [14] hacen uso de técnicas de MDD para ayudar a reducir el coste de complejidad del proceso de desarrollo de videojuegos mediante un solo DSL centrado en el modelado de juegos 2D. Esto les permite elevar el nivel de abstracción del desarrollo de juegos mediante modelos y reducir costos de tiempo y esfuerzo en dicho desarrollo. A pesar de que muestran garantías para la generación de videojuegos, ellos sólo se apoyan en un DSL y discuten que el motor que sostiene el juego sólo consume el código generado en vez de apoyarlo mediante clases que manejen esa información para hacerlo más efectivo y estructurado.

Dobbe [8] ofrece el desarrollo de un DSL independiente del género para diseñar juegos, esto permite trasladar el ámbito del juego a un nivel de abstracción mayor donde contempla varios aspectos para la implantación de un diseño de juego, estos aspectos pueden ser considerados como requisitos principales para el funcionamiento y definición del dominio del DSL utilizando los siguientes aspectos: objetos que comparte el mundo del juego, las interacciones con el jugador, las reglas que rigen la mecánica del juego y la historia que nos presenta. Esto puede ser de gran interés para los diseñadores de juegos pero no aborda más etapas del desarrollo ni dice como se transformarán las especificaciones en la implementación..

2.1 Aplicación de MDA en el desarrollo de videojuegos

La utilización de MDA aplicada al desarrollo de videojuegos permite capturar los conceptos de la jugabilidad de un juego en un modelo PIM. La identificación de estos aspectos permite a los diseñadores expresar cualquier concepto del dominio independiente de la plataforma abstrayéndose de un lenguaje de programación.

Partiendo del ciclo de la jugabilidad de Crawford [5], los videojuegos pueden considerarse sistemas interactivos en el que los jugadores se comunican con el juego a través de acciones de entrada realizadas mediante controladores hardware. Este ciclo se repite continuamente durante la interacción entre los jugadores y el juego.

En la sección 2.1.1 presentamos la perspectiva de la interfaz gráfica de usuario a través de un diagrama de navegación, en la sección 2.1.2 se presenta la perspectiva de control. Por último en la sección 2.1.3 se detalla la perspectiva de las reglas e identidades [11] que forman parte de las vistas que hemos definido en nuestro framework.

2.1.1 Perspectiva de la interfaz Gráfica de Usuario

Para facilitar ciertos aspectos del modelado visual a los diseñadores de juegos, la perspectiva GUI muestra información a los jugadores de cómo está estructurada la navegación de las pantallas del videojuego y cómo se organiza la información en éstas. La figura 1 muestra el meta-modelo de navegación que representa las principales primitivas de desplazamiento entre pantallas. El principal objetivo de esta perspectiva es que el diseñador pueda hacer transiciones entre pantallas representados como nodos a través de transiciones de una pantalla a otra declarando sus respectivos eventos de entrada o salida, tales como eventos del juego, interacciones de control o tiempo.

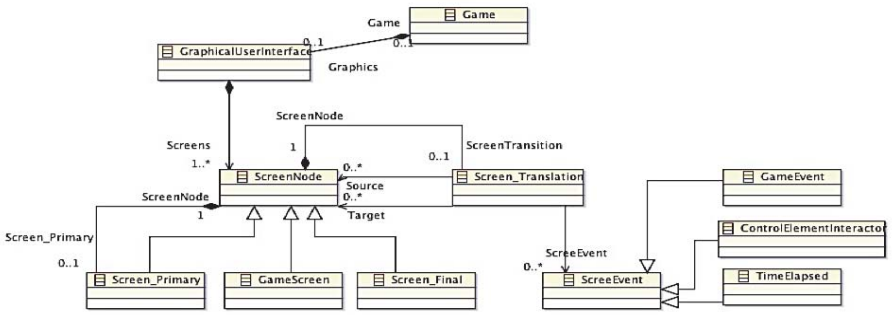


Fig. 1. Meta-modelo de navegación entre pantallas

2.1.2 Perspectiva de Control

El control define cómo se comunican los jugadores con el juego a través de dispositivos hardware controladores. Cada plataforma tecnológica ofrece diversos controladores que permiten a los jugadores distintas interacciones. Todos los controladores ofrecen a los jugadores elementos de control que permiten enviar información al sistema

de juego. Así, se pueden definir algunos conceptos de control independientes de la tecnología, comunes a todos los controladores y plataformas tecnológicas.

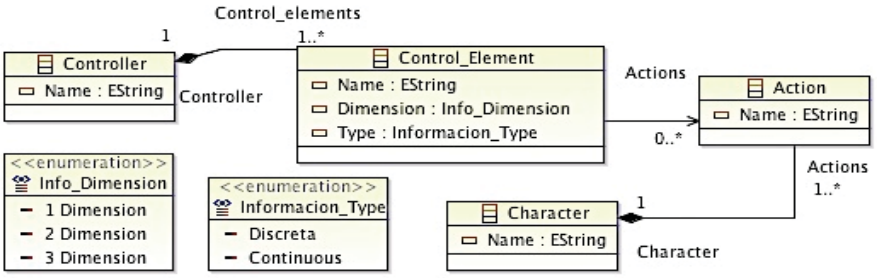


Fig. 2. Meta-modelo de control.

La Figura 2 muestra el meta-modelo de control. Un controlador está formado por elementos de control (botones, joysticks, etc..). Los elementos de control envían información al sistema de juego (los botones envían información 0-dimensional, los gatillos 1-dimensional y los joysticks 2-dimensional). Los jugadores interactúan (pulsan, sueltan, mantienen, etc…) los elementos de control para ejecutar reglas de acción del conjunto de reglas .

2.1.3 Perspectiva de las Entidades y Reglas

En el dominio del desarrollo de juegos, las reglas y las entidades juegan un gran papel para la especificación de la jugabilidad. Especificar qué entidades del juego existen y qué reglas se disparan en el videojuego permite definir de forma precisa la interactividad del juego.

A continuación, se detallan las principales primitivas de diseño para la especificación de la jugabilidad mediante reglas y entidades a través de la figura 3.

Una entidad es un elemento que conforma el mundo y que se comporta según las normas del mismo. Un mundo esta compuesto por entidades con atributos y comportamiento, que además interactúan con el mismo según las normas de la lógica de juego. El comportamiento de una entidad viene dado por las normas definidas.

Una regla es una sentencia declarativa con una o más pre- y post-condiciones que establecen qué condiciones deben satisfacerse antes y después de la aplicación de la regla, respectivamente. Del mismo modo, todas las pre- y post-condiciones se expresan utilizando conceptos previamente definidos en el modelo de estructura para la jugabilidad: acciones, eventos y atributos.

Los atributos representan características propias de la entidad de juego. Los eventos representan sucesos disparados por la entidad de juego que cambian el estado del sistema de juego.

La meta-clase *Outcome* define un tipo especial de evento de resultado que establece la consecución de un objetivo de juego.

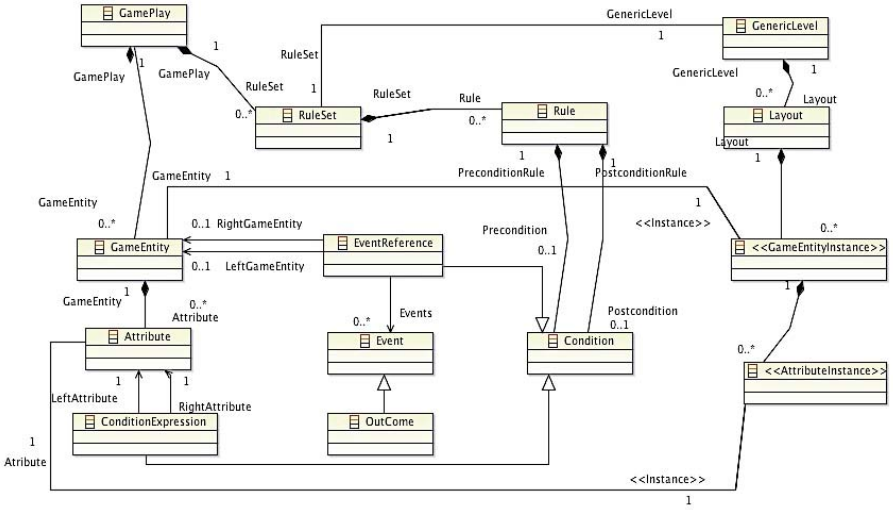


Fig. 3. Meta-modelo de las Entidades y Reglas de un nivel.

3 Perspectiva del Diseño de Niveles

Uno de los objetivos fundamentales de la creación de un videojuego es la forma precisa de cómo se estructura el diseño de los niveles que compone nuestro juego y cómo integrar los aspectos más importantes del comportamiento de la jugabilidad, acción y guión [3]. Cada nivel engloba gran cantidad del contenido de un juego y es una parte que hace depender la evolución de las iteraciones en el desarrollo de un videojuego [9].

El diagrama del diseño de niveles es un modelo visual que facilita a los diseñadores especificar la estructura global de los niveles que contiene el juego, abstrayéndonos del comportamiento del sistema.

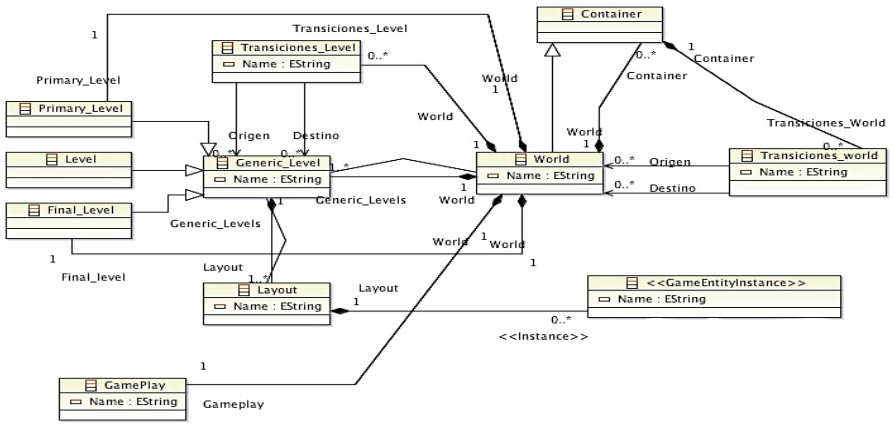


Fig. 4. Meta-modelo del diseño de niveles.

Nótese que en la figura 4, la meta-clase *World* actúa como principal primitiva para la creación de los niveles. Se ha definido un mundo como un contenedor para utilizar una técnica usada por los guionistas y mapeadores de niveles que permite separar varias historias dentro de la narrativa de un juego, permitiendo así, especificar diferentes *gameplays* en los diferentes escenarios del juego. Un claro ejemplo de uso es el utilizado en todas las sagas de Super Mario Bross. Dónde se definen mundos con transiciones entre éstos y se declaran niveles de juego dentro de estos mundos.

Otra característica que nos muestra este meta-modelo es la utilización de las capas que forman un nivel para hacer uso de técnicas de animación en el mundo de los videojuegos. Un *layout* es el conjunto de capas de presentación del juego. Esto permite al diseñador la declaración de tantos *layouts* como capas de representación se quiera. Todas estas "cajas" de diseño representan un conjunto que describen la composición final de la presentación del nivel [4]. La figura 5 (izquierda) muestra la técnica en uso permitiendo al diseñador diferenciar las capas de presentación de un nivel específico para separar en *layouts* los distintos activos que componen el nivel de un juego como son el *background*, *HUD*, *tiles*, etc…

Estas entidades se representan mediante la primitiva *GameEntityInstance* dentro de un *layout*. Son instancias que permiten la creación de un objeto perteneciente a una clase. Su representación en el meta-modelo está especificado mediante estereotipos tal y como se describe en [12].

Para facilitar el modelado visual a los diseñadores de juegos, se ofrece un DSL para especificar un diagrama del diseño de niveles. La figura 5 (derecha) muestra dos mundos que se relacionan entre sí mediante una transición descrita con una línea discontinua. Cada mundo contiene sus niveles correspondientes. Se observa cómo en la descripción de los niveles tenemos una condición de obligación que describe el inicio como un punto negro y un final con dos puntos concéntricos, siendo blanco el circulo del interior [13]. Las transiciones entre los niveles se describen con flechas continuas siendo éstas tanto de origen como de destino. Se observa cómo el nivel 1 avanza al nivel 2 mediante la transición T1, el nivel 2 apunta al nivel 3 mediante la transición T2, el nivel 3 describe dos posibles transiciones, retroceder al nivel 1 o avanzar al nivel 4, este último nivel puede bien retroceder al nivel 2 mediante la transición 2 o finalizar el mundo.

4 Más allá del GamePlay de un juego, un Motor de Tiles.

4.1 Motor de Tiles 2D.

Tratar de separar las principales partes que componen un juego permite profundizar y ampliar en el desarrollo de los conceptos más comunes. La elección de usar motor de *tiles* viene regida por el desarrollo iterativo de videojuegos en 2D. Todos los géneros de juegos RPG, plataformas, *scrollers* … toman como referencia el uso de *tiles*. Ésta no es más que una unidad que representa un gráfico en una área constituida por 2 dimensiones definida como una matriz que contiene las referencias de los objetos instanciados en un nivel del juego. Estas entidades del juego deben representar la

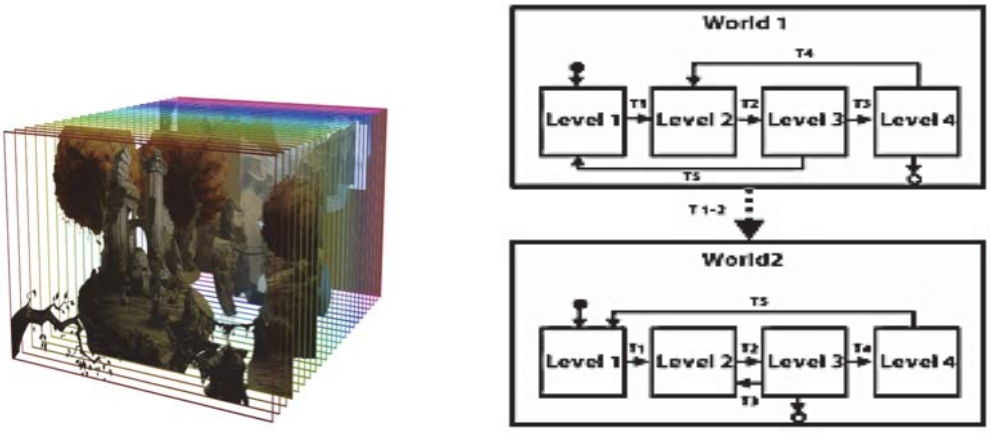


Fig. 5. (Izquierda) Representación gráfica de la composición de los distintos layouts que contiene un nivel del juego "the Whispered World", y (derecha) diagrama del diseño de niveles genérico 2D.

información del tipo de terreno, si es posible o no caminar sobre ella, si causa daño o beneficio al personaje del juego… Este enfoque simple permite a los diseñadores de juegos especificar de forma sencilla la representación de grandes mundos y niveles de juegos basados en *tiles* a través de mapas.

Estas *tiles* generalmente son simples representaciones geométricas como se observa en la figura 6 (izquierda) pudiendo ser extendidas a formas geométricas más elaboradas como hexágonos muy comunes en los juegos de puzles. Su definición viene dictada por una altura y anchura uniforme en todo el mapa.

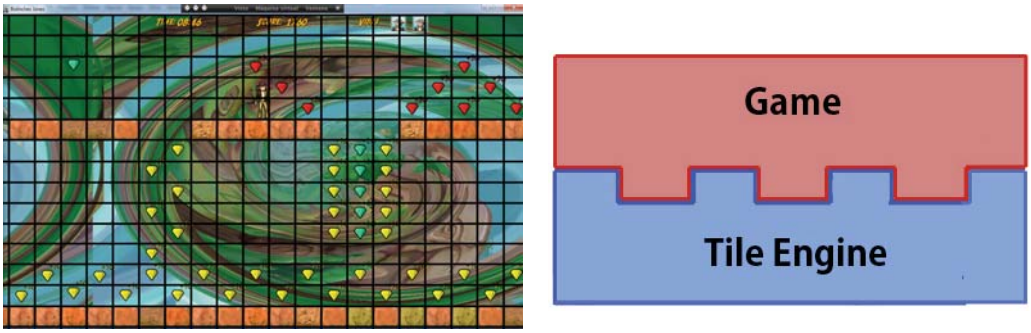


Fig. 6. (Izquierda) Representación de un nivel mapeado mediante *tiles,* y (derecha) esquema del acoplamiento del juego con el motor de *tiles* 2D.

El uso de esta técnica permite separar 2 grandes aspectos en el desarrollo de un videojuego como es la especificación del juego y el motor que lo apoya en tiempo de ejecución, ver figura 6 (derecha). Siguiendo un aproximación MDA usando el modelo PIM, cuando el diseñador especifique el *gameplay*, la GUI, el Control y el Diseño de Niveles del juego es transformado a un PSM llamado *Tile Engine* obteniendo un mo-

delo específico de la plataforma con la especificación del juego que hace uso de los conceptos del motor.

4.2 Modelo Específico de la Plataforma del motor de tiles

A continuación, se muestran las distintas perspectivas de la especificación de la interactividad del juego que han sido capturadas en el PIM y serán transformadas en varias vistas del modelo PSM que representa el motor de *tiles* 2D genérico (figura 7). La relación de trazabilidad entre los conceptos del modelo origen y destino no se pierden. Dado que el modelo PSM recoge características que son propias de la plataforma y que no están presentes en el PIM, el diseñador tendrá que introducir manualmente esta información que es específica de la plataforma. Esta información añadida se detalla en cada una de las perspectivas que se muestran a continuación.

Por simplicidad, se han conservado muchos de los nombres de las meta-clases definidas en el modelo PIM para hacer más sencilla la transformación al PSM.

4.2.1 Perspectiva del Manager de la IGU

GraphicDeviceManager es la primitiva de la interfaz gráfica de usuario que define cómo se administra la navegación (cómo se organiza la información de las diferentes pantallas del juego) y la distribución (qué tipo de información se detalla) de todas las pantallas definidas previamente en el modelo independiente de la plataforma. En esta clase se observa cómo se especifican ciertos atributos dependientes de la plataforma que nos permiten establecer los valores predefinidos del buffer tanto del ancho como el alto de la pantalla para que cuando el diseñador genere *ScreenNodes* tengan todos ellos unos valores preestablecidos.

La clase *ScreenManager* permite al modelo PSM capturar todos aquellos detalles que en la perspectiva de la Interfaz Gráfica de Usuario del modelo PIM se han definido previamente.

La clase *ScreenNode* ahora cuenta con atributos de posicionamiento de coordenadas para que el diseñador pueda especificar la ubicación exacta de la pantalla, pudiendo así, definir dos *GameScreens* para que 2 jugadores puedan compartir la pantalla tanto en horizontal como en vertical en caso necesario. Estos nodos siguen relacionándose entre si mediante transiciones declarando sus respectivos eventos de entrada o salida, tales como eventos propios del juego, interacciones de control o tiempo. La primitiva *ScreenNode* está compuesta por las clases *DisplayPrimitive*, *SpriteFont* y *SpriteBatchScreen*.

La clase *DisplayPrimitive* define qué tipo de información se detalla en la pantalla, permitiendo así, la definición de atributos que se posicionan en el *GameScreens* mediante la primitiva *SpriteFont* que es el diccionario de caracteres que forman estos atributos y está asociado al *Batch* de la pantalla.

La primitiva *SpriteBatchScreen* en la clase que permite dibujar en la pantalla. Está compuesta por la clase *SpriteBatchDrawScreen* que facilita el pintado de *texturas2D* tales como *overlays*, *background*... y la clase *SpriteBatchStringScreen* que permite el pintado de los atributos en modo texto ubicados en la pantalla.

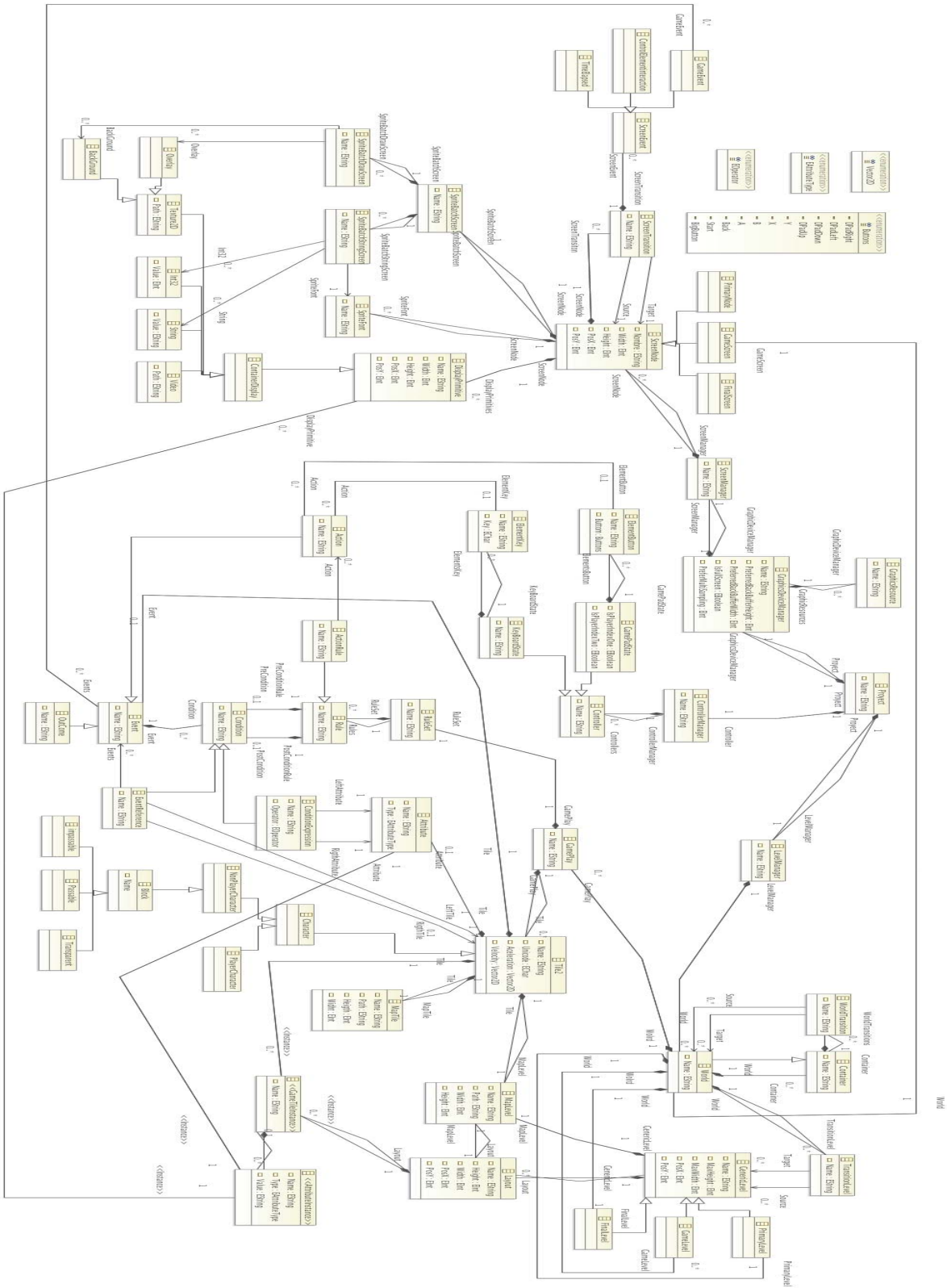


Fig. 7. Meta-modelo específico de la plataforma

4.2.2 Perspectiva del Manager de Control

El manager de control define cómo se comunican los jugadores con el juego a través de dispositivos hardware. Dada la especificación de los controles en el modelo independiente de la plataforma en el que solamente se declara un *controller* como una clase que interactúa con el juego, ahora se trata de especificar mediante la especialización de un teclado o *Pad's* específicos de la plataforma.

La clase *ControllerManager* permite gestionar los controles del juego. Está compuesta por la primitiva *Controller* que transforma la información definida en el modelo PIM en dos posibles tipos de controles específicos.

La clase *GamePadState* es un tipo de control definido como *pad*. En ella el diseñador puede especificar mediante sus atributos si el control pertenece al *player* 1 ó 2. Está compuesta por la clase *ElementButton* que a través de una clase enumeradora llamada *Buttons* especifica todos sus botones para asociarlo a una acción del juego.

La clase *KeyBoardState* en un tipo de control definido como teclado. Está compuesta por la primitiva *ElementKey* que permite especificar qué tipo de tecla va a tener asociado una acción del juego.

4.2.3 Perspectiva del Manager Niveles

La clase *LevelManager* permite gestionar la distribución del diseño del juego procedente de las especificaciones del diseñador en el modelo PIM y está compuesta por las primitivas *World*, *GenericLevel*, *GamePlay* y *TransitionLevel*.

La clase *World* está asociada a transiciones entre mundos que permite la navegación entre éstos pudiendo declarar un mundo como un contenedor de niveles. Todo mundo debe de tener por obligación un nivel primario y final para poder avanzar al siguiente mundo. También esta asociado a la clase *GameScreen* del manejador de pantallas para poder visualizar el mundo con sus niveles.

La clase *GameLevel* permite transformar los niveles que se han creando en el modelo PIM. Se le añaden atributos específicos tales como el largo y ancho máximo del nivel para delimitar el rango de la longitud de las *tiles* y atributos de posicionamiento. Esta compuesta por la clase *Layout* que transforma las clases procedentes del modelo PIM y por la clase *MapLevel* que el diseñador utiliza para definir el diseño del nivel a través de *tiles*. La especificación de un nivel se hace a través de un pequeño documento de texto que el motor de *tiles* transforma en un vista del *Layout* asociada a ese nivel.

La clase *Layout* está compuesta por *GameTileInstance* que son instancias de *tiles*.

4.2.4 Perspectiva de Tiles y Reglas

Todas las entidades declaradas en el modelo PIM se transforman en *tiles* en el modelo específico de la plataforma. Es en clase *Tile* en la que se define el atributo Unicode que va asociado a la clase *MapTile* en la que se obtiene su representación visual. Se especifican ciertos atributos para caracterizar el comportamiento de una *tile* como es su aceleración y velocidad a través de vectores de dos dimensiones.

Un *tile* puede especializarse en un clase *Character* que a su vez especializa en las clases *PlayerCharacter* para indicar el personaje principal del juego y *NonPlayer-Character* haciendo referencia a todos los personajes restantes. Es aquí donde la *tile* puede seguir especializándose siendo la representación de un bloque impasable (el jugador no puede atraversarlo), pasable (el jugador solo puede atravesarla por abajo y situarse encima de ella) y transparente (la *tile* es un elemento con el que el personaje no interactúa).

La primitiva *GamePlay* define el conjunto de reglas asociadas a un mundo y las *tiles* que componen la representación de un nivel.

La clase *RuleSet* está compuesta por reglas que especializan en acciones. Éstas están asociadas a la clase *Action* que interactúan con los elementos de un control.

4.3 Transformación de modelo PSM a código

Los conceptos plasmados en el modelo PSM que representan el juego apoyado en un motor de *tiles* 2D genérico simboliza la descripción del sistema en términos de una plataforma específica. Finalmente, una transformación modelo-texto genera las estructuras de datos y las clases C# necesarias para poder ser integradas con el motor de *tiles*

El motor de *tiles* está implementado mediante *Microsoft XNA Game Studio*, puesto que ofrece un *framework* con simples primitivas específicas para el desarrollo de videojuegos para tres plataformas de juegos distintas como: PC, XBOX 360 y WP7.

5 Conclusiones y trabajos futuros

El objetivo de este artículo es mostrar a través de un modelo PIM diferentes especificaciones de las perspectivas de los videojuegos tales como la interfaz gráfica de usuario, el control, el diseño de niveles, las entidades y reglas que forman el gameplay del juego. También se ofrece la propuesta de un modelo PSM genérico que permite definir la estructura y el comportamiento de un motor de tiles 2D que integra la jugabilidad del juego.

El editor de niveles ha ayudado a especificar la estructura externa de la organización de los niveles de un juego así como las relaciones que tienen entre éstos. El uso de la técnica de animación representada mediante la primitiva *Layouts* ha servido para especificar también la estructura interna de un nivel dando ventajas al diseñador del juego con las diferentes vistas que contiene la composición final de la presentación de un nivel.

La propuesta para especificar un motor de *tile*s 2D mediante MDA ha favorecido el desarrollo de la propuesta de videojuegos en un escenario 2D reutilizando componentes y separando aspectos de un juego como es la especificación del *gameplay* en distintas perspectivas. Permite a nuestro motor de *tiles* 2D asentar todos estos conocimientos dando soporte a aquellos conceptos que no se alcanzaban en la especificación del modelo PIM.

La definición del motor de *tiles* 2D genérico en un modelo PSM nos favorece para

poder enfocar el desarrollo de videojuegos en el mercado de la industria con el auge de las nuevas tecnologías. De esta forma se puede especificar el *gameplay* de un juego a un nivel muy alto de abstracción y orientarlo a un motor específico de la plataforma para dar soporte a la tecnología que se desee.

Como trabajos futuros, se incluye la definición de otras perspectivas del juego como la historia o la inteligencia artificial que permitirán especificar juegos más complejos, la posible implementación del motor de *tiles* 2D en un lenguaje de desarrollo como Java para abordar prácticamente el mercado de dispositivos móviles de última generación y *Tablets*, también utilizar el conocimiento existente en el uso de líneas de productos software para poder obtener una reutilización de software más estructurada y eficaz.

Agradecimientos. Queremos agradecer la asistencia del Ministerio de Ciencia y Tecnología ya que la investigación se ha financiado como parte del proyecto MULTIPLE, con referencia TIN2009-13838.

Referencias

1. Andre W.B. Furtado, Andre L.M. Santos, and Geber L. Ramalho, Federal University of Pernambuco and Eduardo Santana de Almeida, Federal University of Bahia: Improving Digital Games Development with Software Product Lines (2011).
2. Emanuel Montero y José A. Carsí: MDA y Desarrollo de videojuegos, JISDB 2011.
3. Sanchez Crespo, D. : Diseño de niveles, pp, 65-74 .- Hacia la excelencia.
4. Rod tejada . : http://rodtejada.wordpress.com/author/rodtejada/page/2/ - Layout
5. Crawford, C.: On Game Design, pp 76-78, New Riders Publishing (2003).
6. Kleppe, A., Warmer, J., Bast, W.: MDA Explained: The Model Driven Architecture: Practice and Promise. Addison Wesley (2003).
7. Gregory, J.:Game Engine Architecture, A K Peters (July 10, 2009).
8. J. Dobbe, "A Domain-Specific Language for Computer Games," MSc dissertation, Dept. of Software Technology, Delft Univ. of Technology, 2007.
9. E. Byrne. Game Level Design. Charles River Media Boston, 2005.
10. Fullerton, T., Swain, C., Hoffman, S.: Game Design Workshop: Designing, Prototyping, and Playtesting Games, CMP Books (2004).
11. Montero Reyno E: "Desarrollo de Juegos Dirigido por Modelos: Multi-Modelo para la Especificación de la Jugabilidad, Interfaz Gráfica de Usuario y Control de Videojuegos". Trabajo de Fin de Master, Departamento de Ingeniería de Software y Sistemas de Información. UPV 2008.
12. Colin Atkinson, Thomas Kuhne: Model-Driven Development: A Metamodeling Foundation. IEEE Software 20(5): 36-41 (2003).
13. Dormans, J. (2011c). Level Design as Model Transformation: A Strategy for Automated Content Generation. In Proceedings of the Foundations of Digital Games Conference, Bordeaux France, June 2011

14. F.E. Hernandez and R.R. Ortega, “Eberos GML2D: A Graphical Domain-Specific Language for Modeling 2D Video Games,” Proc. 10th SPLASH Workshop on Domain-Specific Modeling, Aalto-Print, 2010;
15. Sid Meier. : GDC 2012: Sid Meier on how to see games as sets of interesting decisions, More Interesting Decisions.

Introducing Safety Requirements Traceability Support in Model-Driven Development of Robotic Applications*

Pedro Sánchez[1], Diego Alonso[1], Francisca Rosique[1], Bárbara Álvarez[1], and Juan A. Pastor[1]

División de Sistemas e Ingeniería Electrónica (DSIE)
Universidad Politécnica de Cartagena, 30.202, Cartagena, España
{pedro.sanchez, diego.alonso}@upct.es

Summary

Teleoperated service robots are used to perform hazardous operations in hostile environments such as nuclear reactors, space missions, warehouses, etc. Since they have to interact with both the environment and human operators, it is essential that they be so designed as to involve no risk to the operators, the environment, or the robot itself. Where it is impossible to eliminate the risk, this at least must be limited.

The work described in this article was developed in the context of the European Union V Framework Programme EFTCoR project (*Environmental Friendly and Cost-Effective Technology for Coating Removal*), which addressed the development of a solution to the problem of retrieval and confinement of sub-products from ship maintenance operations. Given the experience of the DSIE research group in both the design of component-based software applications for tele-operated service robots [1], and the combined use of safety standards (like ANSI/RIA 15.06-1999 and European Standard EN 61508:2001) with specific methodologies for safety systems development (like Rapid Object-Oriented Process for Embedded Systems, ROPES) [2], we decided to develop an integrated development framework.

The Model-Driven Software Development (MDSD) approach can provide a suitable theoretical and technological support for integrating different facets of safety critically system in a global development framework. To implement a system that assures safety, a component-based approach, and traceability, there are a number of conceptual and development (technical, tools, etc.) requirements that have to be met. The absolutely essential ones are basically:

1. An integrated framework of safety management requirements.
2. A catalogue of architecture solutions for these requirements.
3. A framework for automatic generation of code for different implementation platforms.
4. Support for traceability of the various artifacts involved in the development.
5. Tools which enable fully automated handling of all the information involved in the process.

* This work has been partially supported by the Spanish CICYT Project EXPLORE (ref. TIN2009-08572), the Séneca Project MISSION-SICUVA (ref. 15374/PI/10), and the Spanish MEC FPU Program (grant AP2009- 5083).

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 519-520, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

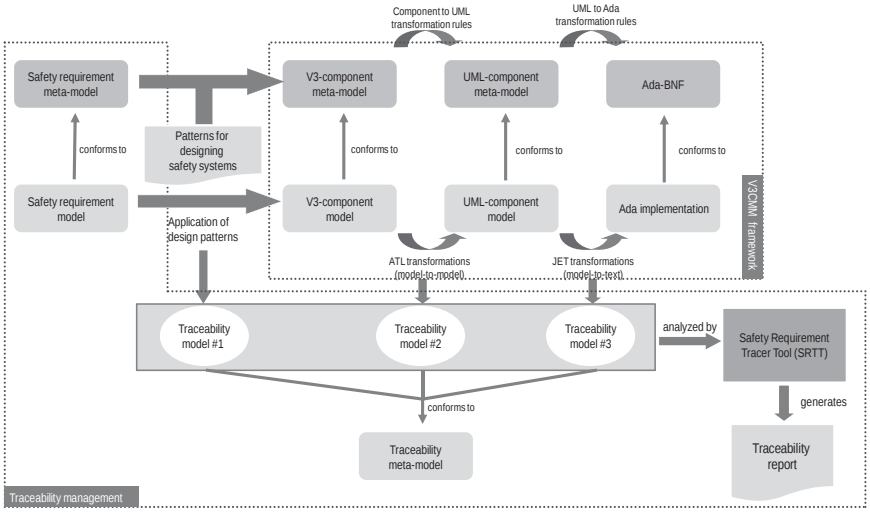


Fig. 1. General MDSD development framework for component-based applications, completed with safety requirements traceability management.

This article presents an integrated development environment that furnishes these resources (see Fig. 1), and demonstrates the viability of the approach on the basis of experience acquired in the development of an industrial robotic system with safety requirements, which represents a relevant case study in synergy between the Robotics and Software Engineering domains. While MDSD is a well-known Software Engineering paradigm and Robotics is a discipline with a long tradition, as far as we know, there is no experience on Robotics that uses in an extensive way the MDSD approach. Nevertheless, there are tremendous opportunities to start with reusable and semantically well-defined designs of complex robotic software systems. The work described in this article offers developers an MDSD environment which: (1) promotes the reuse of the software artifacts generated in the safety systems construction process, and (2) considers the traceability of these artifacts in order to enhance the quality of the systems, since it facilitates both modification and analysis of the impact of the changes on the safety requirements. It is worth noting that these results can be easily extrapolated to other domains different to robotics, since the software artifacts developed and used are indeed domain independent.

References

1. A. Iborra, D. Alonso, F. Ortiz, J. Franco, P. Sánchez, and B. Álvarez: Design of service robots. IEEE Robotics and Automation Magazine, Special Issue on Software Engineering for Robotics, vol. 16, no. 1, pp. 24–33, 2009.
2. D. Alonso, P. Sánchez, F. Ortiz, J. Pastor, B. Álvarez, and A. Iborra: Experiences developing safe and fault-tolerant tele-operated service robots, a case study in shipyards. In Service robot applications. Ed. In-Tech, 2008, pp. 159–182.

Un repositorio NoSQL para acceso escalable a modelos

Javier Espinazo Pagán[1], Jesús Sánchez Cuadrado[2] y Jesús García Molina[1]

[1] Universidad de Murcia, España jespinazo@um.es, jmolina@um.es
[2] Universidad Autónoma de Madrid, España jesus.sanchez.cuadrado@uam.es

Abstract. La aplicación de la Ingeniería Dirigida por Modelos (MDE) en sistemas de escala industrial requiere de complejos modelos que pueden llegar a ser muy grandes y que deben ser almacenados de tal forma que puedan ser manipulados por aplicaciones cliente sin necesidad de ser cargados por completo. En este artículo presentamos Morsa, un repositorio para la manipulación escalable de modelos grandes usando carga bajo demanda, guardado incremental y un API de consultas; la persistencia de modelos es llevada a cabo por una base de datos NoSQL.

Keywords: persistencia de modelos, repositorio de modelos, escalabilidad, modelos grandes

1 Introducción

La creciente madurez de la Ingeniería Dirigida por Modelos (MDE) está promoviendo su adopción por grandes compañías [1][2] que se benefician en términos de productividad, calidad y reuso. Sin embargo, la aplicación de MDE en este contexto requiere de herramientas de escala industrial que puedan operar con modelos muy grandes y complejos. Una operación básica de dichas herramientas es la persistencia de modelos y su acceso, debiendo satisfacer dos requisitos esenciales: escalabilidad e integración.

Uno de los principales obstáculos para la adopción de MDE en la industria es la *escalabilidad* de las herramientas a la hora de acceder a modelos grandes. Como se dice en [3], *"la escalabilidad es lo que echa para atrás a un número importante de potenciales usuarios"*. Un enfoque para abordar la escalabilidad es la partición de modelos mediante mecanismos de modularización propios del lenguaje de modelado [3]. Sin embargo, la complejidad de los modelos grandes dificulta su particionado automático en fragmentos que sean fácilmente accesibles [4], por lo que es imprescindible tener una solución de persistencia de modelos escalable. Para la serialización (es decir, persistencia) de modelos se suele usar el formato XMI (XML Metadata Interchange) [5], el cual requiere la carga completa de un modelo para poder acceder a cualquiera de sus elementos, lo cual puede desbordar la memoria del cliente si dicho modelo es demasiado grande. Los

Este trabajo ha sido financiado por el Ministerio de Ciencia de España (proyecto TIN2009-11555) y la Fundación Séneca (beca 14954/BPS/10).

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 521-534, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

repositorios de modelos están surgiendo como una alternativa a XMI a la hora de manejar grandes modelos. Una preocupación que surge cuando las aplicaciones cliente acceden a modelos almacenados es la *integración*. La integración entre una solución de persistencia y un cliente debe ser *transparente*, esto es, debe ser conforme con la interfaz definida por el framework de modelado (p.ej. la interfaz *Resource* de EMF). Además, es conveniente que dicha solución de persistencia no imponga ningún preprocesado de (meta)modelos previo a su carga o guardado, como por ejemplo la generación de código específico.

En este artículo presentamos Morsa, un repositorio de modelos inspirado en las bases de datos de documentos NoSQL cuya finalidad es conseguir escalabilidad e integración transparente. El problema de la escalabilidad es abordado usando mecanismos de carga bajo demanda y guardado incremental usando una caché de objetos configurable en base a diferentes políticas; se ha desarrollado un API de consultas para aprovechar al máximo la carga bajo demanda. El problema de la integración transparente se soluciona implementando la interfaz del lenguaje de modelado. Hemos desarrollado un prototipo para EMF [8] que usa MongoDB [9] como base de datos NoSQL y que se integra transparentemente con herramientas como los lenguajes de transformación de modelos.

El resto del artículo se estructura de la siguiente forma: la Sección 2 presenta el concepto de persistencia de modelos y el movimiento NoSQL; la Sección 3 introduce el caso de estudio; la Sección 4 muestra el diseño arquitectónico y de datos del repositorio y su implementación e integración en EMF; la Sección 5 explica la carga bajo demanda; la Sección 6 muestra el API de consultas; la Sección 7 comenta el trabajo relacionado y, por último, la Sección 8 muestra nuestras conclusiones y trabajo futuro.

Una versión anterior de Morsa fue presentada en el congreso MODELS 2011 [10], por la cual fuimos invitados a enviar una extensión a la revista SoSyM. Este artículo introduce una versión más actual de Morsa con un rediseño arquitectónico, algorítmico y de datos. Los algoritmos de guardado, actualización y borrado y la evaluación del repositorio son abordados en el artículo enviado a SoSyM. Por otro lado, se presenta un API de consultas nunca antes publicada.

2 Marco conceptual

Esta sección comenta la representación de modelos como grafos de objetos y da una breve introducción a las bases de datos NoSQL.

2.1 Modelos como grafos de objetos

El enfoque de persistencia de modelos presentado en este artículo se basa en el hecho de que los modelos son grafos de objetos, por lo que a lo largo del artículo se usarán algunos términos de teoría de grafos que se definen a continuación. Dado un objeto (esto es, un elemento de un modelo): un *ancestro* es un objeto que (in)directamente lo contiene; un *descendiente* es un objeto (in)directamente contenido por él; un *hijo* es un objeto directamente contenido por él; un *padre* es el objeto que lo contiene directamente; un *hermano* es un objeto con el que

comparte el mismo padre; su *anchura* es su posición dentro de la lista que lo contiene y finalmente su *profundidad* es la cantidad de ancestros que tiene. Finalmente, un *subgrafo* (es decir, una partición de un modelo) es un grafo cuyos nodos y arcos son un subconjunto del grafo dado y un *objeto raíz* es un objeto que no tiene ancestros.

La figura 1 muestra las relaciones de contención de un modelo, las cuales usaremos para ilustrar los conceptos que han sido introducidos: (i) `Obj1` es el elemento raíz, por lo que su profundidad es 0; (ii) los ancestros de `Obj6` son `Obj4`, `Obj3` y `Obj1`, por lo que su profundidad es 3; (iii) `Obj5` es hermano de `Obj4` y su anchura es 2; (iv) los descendientes de `Obj4` son `Obj6`, `Obj7`, `Obj8` y `Obj9` (se podría formar un subgrafo con estos objetos); (v) el padre de `Obj4` es `Obj3` y (vi) los hijos de `Obj4` son `Obj6` y `Obj7`.

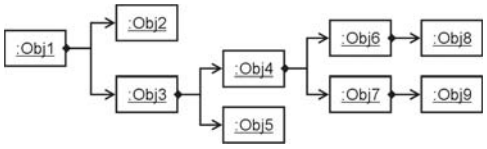


Fig. 1. Ejemplos de relaciones entre elementos de un modelo

2.2 Persistencia de modelos

La *persistencia de modelos* es un servicio que suele ser proporcionado por los frameworks de modelado como EMF a través de la definición de interfaces de persistencia (p.ej. *Resource* en EMF), las cuales permiten a las aplicaciones cliente acceder a los modelos almacenados a través de cinco operaciones básicas:

- *Carga*: un modelo o partición de un modelo es transferido desde la solución de persistencia a la memoria del cliente. Si todo el modelo es reconstruido, la carga es completa; si no, la carga es parcial.
- *Guardado*: un modelo o partición de modelo es transferido desde la memoria del cliente a la solución de persistencia. Si todo el modelo es guardado de una vez, el guardado es completo; si no, el guardado es incremental.
- *Actualización*: un modelo o partición de modelo que ya está almacenado es modificado en la memoria del cliente y transferido al almacenamiento.
- *Borrado*: un modelo o partición de modelo es eliminado del almacenamiento.
- *Consulta*: un conjunto de elementos del modelo que satisfacen una condición dada son transferidos a la memoria del cliente desde el almacenamiento.

Una solución de persistencia proporciona *integración transparente* cuando las aplicaciones cliente pueden acceder a ella usando la interfaz de persistencia definida por el framework de modelado sin ningún tipo de pre o post procesado tal como modificar los modelos o metamodelos para añadir información

de persistencia o generar código específico para la persistencia a partir de los metamodelos. Por ejemplo, XMI no requiere la generación de ninguna clase Java específica para un metamodelo EMF porque usa objetos dinámicos, que se construyen de forma genérica en tiempo de ejecución.

2.3 El movimiento NoSQL

Como se mencionó en la introducción, proponemos un repositorio de modelos cuyo modelo de datos está inspirado en el paradigma NoSQL de las bases de datos de documentos. Las aplicaciones de bases de datos para dominios como la búsqueda textual en la web o el proceso de flujos de datos han demostrado recientemente que los SGBDs relacionales no son adecuados para los requisitos (distribución, escalabilidad, etc.) de dichas aplicaciones [11], por lo que surgen nuevos paradigmas como alternativas a éstos, los cuales se engloban bajo el término *NoSQL* [12]. La principal diferencia entre las bases de datos NoSQL y las relacionales es que, mientras las últimas aseguran las propiedades ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), las primeras sólo consideran la consistencia y no siempre, ya que algunas bases de datos se centran en otras propiedades como la disponibilidad y la tolerancia a particiones. Nuestro enfoque está inspirado en las *bases de datos de documentos* como MongoDB [9], las cuales encapsulan pares clave-valor en estructuras denominadas documentos, proporcionando datos complejos sin necesidad de usar esquemas. Otros paradigmas NoSQL son los almacenamientos clave-valor [13] y las bases de datos de columnas [14].

3 Caso de estudio

Para ilustrar el diseño de nuestro enfoque usaremos un caso de estudio basado en el caso de estudio de ingeniería inversa del concurso Grabats 2009 [15]. Este caso consiste en ejecutar una consulta sobre cinco grandes modelos que representan código fuente Java; dicha consulta debe obtener todas las clases que declaren un método público y estático cuyo tipo de retorno sea la propia clase. La Figura 2 muestra de forma simplificada el subconjunto del metamodelo *JDTAST* usado para las consultas. La información sobre los modificadores y los tipos de retorno se especifica en el paquete `DOM`, pero no hay una referencia explícita que vaya desde el tipo de retorno de un método (objeto `Type`) hasta la declaración de dicho tipo (objeto `TypeDeclaration`), por lo que la asociación de dichos objetos debe ser hecha por nombre. La consulta consiste en seguir estos pasos para cada objeto `TypeDeclaration` del modelo:

1. Obtener el *fullyQualifiedName* del objeto `Name` referenciado por la relación *name*.
2. Encontrar al menos un objeto `MethodDeclaration` referenciado por la relación `bodyDeclarations` tal que: (i) referencie a un objeto `Type` mediante su relación *returnType* y (ii) ese objeto `Type` tenga un objeto `Name` referenciado por su relación *name* tal que (iii) el atributo *fullyQualifiedName* de éste sea igual al *fullyQualifiedName* obtenido en el paso 1.

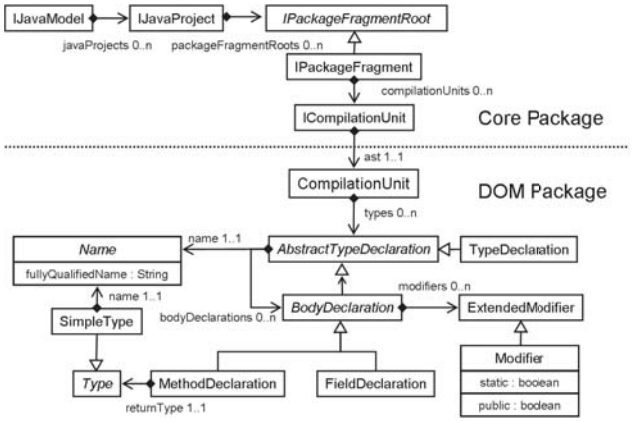


Fig. 2. Simplificación del metamodelo *JDTAST* para el concurso Grabats 2009

4 Diseño e implementación

Esta sección explica los diseños arquitectónico y de datos de Morsa y su implementación e integración con EMF. Morsa tiene una arquitectura cliente-servidor y su diseño de datos está inspirado en el paradigma NoSQL de bases de datos de documentos. Ambos aspectos del diseño han sido concebidos para conseguir *escalabilidad* en el cliente e *integración transparente*.

4.1 Diseño arquitectónico

La arquitectura de Morsa consiste en un lado cliente y un lado servidor, como se muestra en la Figura 3. El primero se encuentra albergado en la máquina cliente (la que ejecuta la aplicación cliente), y el segundo está albergado en una máquina remota (aunque podría ser la misma). El *lado cliente* de Morsa proporciona *integración transparente* gracias a un manejador (`MorsaDriver`) que implementa la interfaz de persistencia del framework de modelado, permitiendo que las aplicaciones cliente manipulen modelos de forma estándar. La carga parcial eficiente de modelos grandes es proporcionada a los clientes mediante un mecanismo de carga bajo demanda, consiguiendo *escalabilidad* en éstos [4]. Este mecanismo depende de una caché de objetos (`ObjectCache`) que guarda los elementos del modelo que han sido cargados y gestiona el uso de memoria; la caché es gestionada por una política de reemplazo configurable (`CachePolicy`). El lado cliente se comunica con el lado servidor usando un adaptador (`MorsaBackend`) que lo abstrae de la base de datos usada. La manipulación de los objetos del repositorio y de las consultas a la base de datos usa un codificador (`MorsaEncoder`). El *lado servidor* de Morsa consta de una base de datos que proporciona el almacenamiento físico de modelos.

4.2 Diseño de datos

El modelo de datos ha sido diseñado para representar los objetos almacenados en el repositorio en el *lado cliente* de Morsa de forma independiente de la base

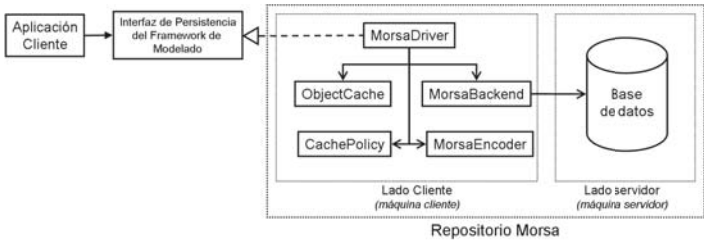


Fig. 3. Arquitectura del repositorio Morsa

de datos utilizada. Tanto los modelos como los metamodelos pueden ser vistos como grafos cuyos nodos son los elementos del modelo y cuyos arcos son las relaciones entre ellos (ver Sección 2.1). Un (meta)modelo (o partición de uno) se representa en Morsa como una colección de `MorsaObject`s conectados a través de `MorsaReference`s. A estas colecciones se las conoce como `MorsaCollection`s y tienen un identificador (p.ej. la URL del metamodelo). La Figura 4 muestra un ejemplo de representación de un modelo en el repositorio Morsa. En el lado izquierdo se muestra una instancia del metamodelo *JDTAST* (ver Figura 2), esto es, un modelo, y en el lado derecho, un conjunto de de `MorsaObject`s representa tanto el modelo como la parte del metamodelo referenciada por los elementos de éste. Las líneas continuas representan relaciones entre elementos y las discontinuas, relaciones de instancia entre los objetos y sus metaclases.

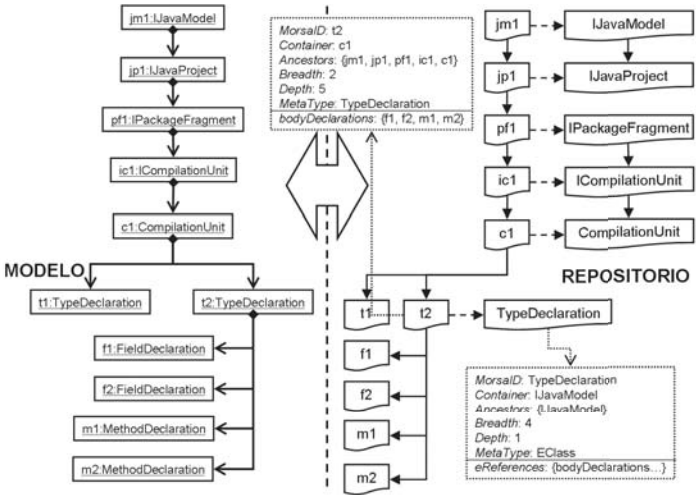


Fig. 4. Representación en el repositorio del caso de estudio

Cada `MorsaObject` representa un elemento del modelo y está compuesto por un conjunto de pares clave-valor que codifican sus características. La clave es el

nombre de la característica y el valor es un valor primitivo si la característica es un atributo o una `MorsaReference` si es una relación. Un `MorsaObject` también contiene un descriptor con metadatos para su identificación, consulta y optimización:

- *MorsaID*: identificador único para el repositorio y dependiente de la base de datos (p.ej. un UUID).
- *Metatype*: `MorsaReference` al `MorsaObject` que representa la metaclase del elemento.
- *Container*: `MorsaReference` al `MorsaObject` que que representa al elemento del modelo que contiene a éste.
- *Ancestors*: lista de `MorsaReference`s a los `MorsaObject`s que representan los ancestros del elemento.
- *Breadth* y *Depth*: anchura y profundidad del elemento, respectivamente, como se definieron en la Sección 2.1.

El *MorsaID* es una característica crucial ya que permite a la `ObjectCache` identificar unívocamente los objetos que han sido cargados en el lado cliente. *Metatype* permite la inferencia de las características de un objeto en el lado cliente. El resto de características representan la estructura del grafo de objetos y son usadas en la carga bajo demanda. Una `MorsaReference` está compuesta del *MorsaID* del elemento referenciado y del identificador de la `MorsaCollection` que lo contiene, esto es, el (meta)modelo al que pertenece. Las cajas discontinuas de la Figura 4 muestran la estructura interna de los `MorsaObject`s que representan los elementos `t2` y `TypeDeclaration`. Los valores para las características *MorsaID*, *Container*, *Ancestors* y *Metatype* han sido simplificados al nombre del elemento por razones de legibilidad.

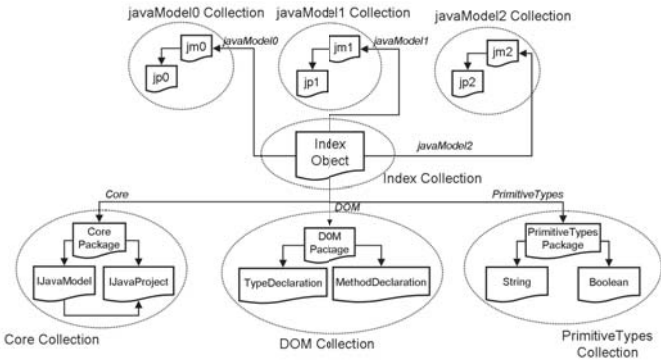


Fig. 5. Colecciones contenidas en el repositorio para los paquetes del metamodelo *JDTAST* y tres modelos

Una `MorsaCollection` especial denominada *colección índice* (*index collection*) contiene el *objeto índice* (*index object*), que es un `MorsaObject` único cuyas

claves son los identificadores de los (meta)modelos almacenados en el repositorio (p.ej. URIs para EMF) y cuyos valores son `MorsaReferenece`s a los objetos raíz de dichos (meta)modelos; estos objetos son usados por el `MorsaDriver` para acceder a los (meta)modelos. Por cada paquete de cada metamodelo se crea una `MorsaCollection`. La Figura 5 muestra cómo el objeto índice (*index object*) referencia los objetos raíz de los tres paquetes (`Core`, `DOM` y `PrimitiveTypes`) definidos en el metamodelo de la Figura 2 y a los de tres modelos de ejemplo. Nótese que el objeto índice apunta al elemento `jm1`, el cual es la raíz del modelo `javaModel1` mostrado en la Figura 4.

4.3 Integración e implementación

Hemos implementado un prototipo [8] de Morsa que se integra de forma transparente con EMF y que usa una base de datos de documentos como almacenamiento de modelos. La integración de Morsa con EMF se logra gracias al `MorsaDriver`, que implementa la interfaz de persistencia del framework (*Resource*). Además, la persistencia de un modelo en Morsa no necesita de ningún pre o post procesado ya que los metamodelos se almacenan de forma transparente si no se encuentran ya en el repositorio al almacenar el modelo; tampoco se requiere la creación de clases Java específicas del modelo. La persistencia puede ser configurada usando los parámetros de los métodos de la interfaz *Resource*. Hemos escogido MongoDB [9] como el motor de base de datos para nuestro prototipo por sus consultas dinámicas, programación JavaScript en el lado del servidor y su comunicación de objetos ligera basada en BSON [16]. Por estar inspirados en las bases de datos de documentos (ver Sección 4.2), la mayoría de los conceptos de Morsa pueden ser representados directamente en MongoDB: los `MorsaObject`s son `DBObject`s de MongoDB (es decir, objetos BSON), las `MorsaCollection`s son `DBCollection`s (esto es, colecciones de documentos) y los *MorsaID*s son `ObjectId`s.

5 Carga de modelos

Hemos considerado tres escenarios para la manipulación de modelos grandes: *carga completa*, *carga bajo demanda simple* y *carga bajo demanda parcial*. Los escenarios de carga bajo demanda ha sido abordados usando una *caché de objetos* que asocia cada objeto cargado con su *MorsaID*.

5.1 Carga completa

Considérese un modelo que es pequeño o mediano, por lo que cabe en la memoria de la aplicación cliente. Si todo el modelo va a ser recorrido, sería conveniente cargarlo de una sola vez ya que se ahorrará tiempo en la comunicación con el almacenamiento persistente. Llamamos a este escenario *carga completa* y es la forma en la que EMF trabaja con ficheros XMI. El algoritmo de carga completa de Morsa simplemente obtiene todos los `MorsaObject`s de un modelo siguiendo sus relaciones de contención. Los metamodelos siempre se cargan por completo y se conservan en memoria por razones de eficiencia y porque son relativamente pequeños comparados con los modelos.

5.2 Carga bajo demanda

Considérese tanto un modelo que es demasiado grande como para caber en la memoria de la aplicación cliente como un modelo que cabe en la memoria pero del que sólo una parte va a ser recorrida. Una solución eficiente para ambos casos sería cargar sólo los objetos necesarios conforme hagan falta y descargarlos cuando ya no la hagan para ahorrar memoria. Este escenario se denomina *carga bajo demanda*. Definimos dos estrategias de carga bajo demanda: simple y parcial. Un algoritmo de *carga bajo demanda simple* obtiene los objetos de la base de datos de uno en uno. Este comportamiento es adecuado cuando los objetos a los que se va acceder no están directamente relacionados entre sí y la memoria consumida es más importante que el tiempo de consultar la base de datos. La caché resultante contendrá los objetos que han sido recorridos.

Un algoritmo de *carga bajo demanda parcial* obtiene del repositorio un subgrafo de objetos calculado a partir de un objeto raíz dado: para dicho objeto raíz, su subgrafo contiene todos sus descendientes dentro de unos determinados valores de profundidad y anchura. Por ejemplo, considérese que en el modelo mostrado en la Figura 6(a) los objetos `jm1` y `jp1` ya han sido cargados y que se solicita la carga de `pf1` con una profundidad y anchura máximas de 4 y 2 para el subgrafo a obtener, respectivamente. Como el valor de *Depth* de `pf1` is 2, la profundidad máxima del subgrafo es 6. El subgrafo estará formado por lo tanto por los objetos `pf1`, `ic1`, `ic2`, `c1`, `c2`, `t1`, `t2`, `t4`, `t5`, `m1`, `m2`, `f1` y `m4`, pero no incluirá a `t3`, `f2` ni `t6` porque su valor de *Breadth* es 3 (mayor que 2). Nótese que `m3` no ha sido cargado ya que, aunque tiene una profundidad de 6 y una anchura de 1, su padre (`t3`) no forma parte del subgrafo. Este comportamiento es adecuado cuando todos los objetos relacionados con el objeto solicitado van a ser recorridos y la eficiencia en memoria es menos importante que el consumo de tiempo en la comunicación. La caché resultante contendrá tanto los objetos ya recorridos como los que se espera recorrer (ver Figura 6(c)).

El algoritmo de carga bajo demanda se invoca cuando se solicita un elemento del modelo que no está en la memoria del cliente (esto es, en la `ObjectCache`); la solicitud puede ser explícita por la aplicación cliente o implícita cuando una relación se navega y el elemento referenciado no está en la memoria del cliente. Nuestro algoritmo de carga bajo demanda es el siguiente:

1. Se solicita un elemento del modelo. El `MorsaDriver` solicita la obtención del correspondiente `MorsaObject` al `MorsaBackend`.
2. Se crea un nuevo elemento, completando sus atributos con los valores guardados en el `MorsaObject` y sus relaciones con *proxies* que permiten la carga bajo demanda de los elementos referenciados. Estos proxies son objetos especiales que tienen la misma estructura que los elementos del modelo, pero que no contienen valores. En su lugar, albergan una URI que contiene una `MorsaReference` que permite su resolución por parte del repositorio. Cuando un proxy es resuelto se convierte en un elemento del modelo con todas sus características completas.
3. El nuevo elemento del modelo y sus proxies son guardados en la `ObjectCache`, relacionándolos con sus correspondientes *MorsaID*s. En caso de carga bajo

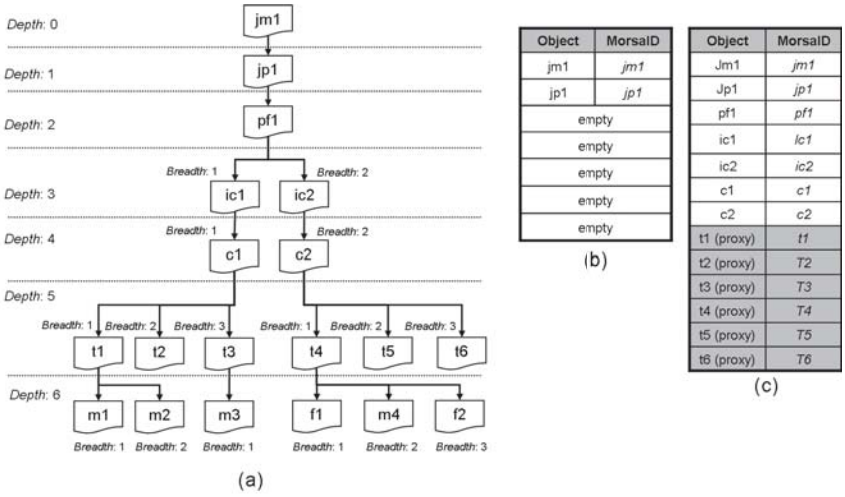


Fig. 6. Carga bajo demanda parcial para el caso de estudio: a) modelo b) caché de objetos antes y c) después de la carga

demanda simple, ir al paso 4. En caso contrario, se solicita al `MorsaBackend` la obtención de todos los objetos del subgrafo definido; éste usa la característica *Ancestors* para calcular los descendientes del objeto solicitado y filtra los resultados obtenidos usando sus atributos *Depth* y *Breadth*. Cada uno de dichos objetos es cargado ejecutando los pasos 1 y 2 de este algoritmo.

4. Si la caché se sobrecarga algunos objetos de la misma deben ser descargados como se explica más adelante.
5. El nuevo elemento del modelo es devuelto a la aplicación cliente, que puede usarlo como un elemento cualquiera más.

El límite de tamaño de la caché se configura como una cantidad de objetos, pero es un límite blando ya que algunos frameworks de modelado como EMF imponen que todo elemento tenga sus relaciones directas completas, es decir, los valores de éstas deben ser rellenados, ya sea como proxies o como elementos propiamente dichos. Por ejemplo, considérese de nuevo el modelo de la Figura 6(a): los elementos `jm1` y `jp1` ya han sido cargados y almacenados en la caché, cuyo tamaño es de 7 objetos como se muestra en la Figura 6 (b). Se solicita la carga bajo demanda parcial de `pf1` con una profundidad y anchura de 2 para el subgrafo, por lo que `pf1`, `ic1`, `ic2`, `c1` y `c2` serán cargados y almacenados en la caché. Sin embargo, como el framework impone la completitud de las relaciones directas, cuando `c1` y `c2` son cargados, sus hijos `t1..t6` también deben ser representados como proxies y almacenados en la caché, sobrecargándola hasta un tamaño de 13 elementos como se muestra en la Figura 6 (c). Siempre que la caché se sobrecargue, los elementos sobrantes deben ser *descargados*, lo que implica degradarlos a proxies, esto es, vaciar todas sus características. Un proxy

ocupa menos memoria que un elemento del modelo y puede ser descartado por el lenguaje anfitrión si no es referenciado por ningún otro elemento.

5.3 Políticas de reemplazo de caché

Una *política de reemplazo de caché* (object `ObjectCache`) selecciona los elementos que deben ser descargados de la memoria del cliente cuando la caché se sobrecarga. Hemos considerado cuatro políticas de reemplazo: *FIFO (Primero en Entrar, Primero en Salir)*, *LIFO (Último en Entrar, Primero en Salir)*, *LRU (Menos Recientemente Usado)* y *LPF (Primero la Partición Mayor)*. Las tres primeras son algoritmos bien conocidos en áreas como los sistemas operativos. En el ejemplo de la Figura 6, una política LIFO descargará los objetos `t1`...`t6`, mientras que una política FIFO o LRU descargaría `jm1`, `jp1`, `pf1`, `ic1`, `ic2` y `c1`. Una política LPF descargaría todos los elementos que conforman la mayor partición del modelo contenida en la caché, lo cual es útil cuando se recorre el modelo sin seguir un orden específico. En el prototipo actual el usuario escoge la política de reemplazo a usar, aunque podría ser escogida automáticamente por el `MorsaDriver` mediante análisis de los (meta)modelos y patrones de acceso.

6 Consulta de modelos

Como se introdujo en la Sección 2.2, una de las operaciones básicas involucradas en la persistencia de modelos es la *consulta*, que suele ser soportada por un *lenguaje de consulta*. Un lenguaje de consulta está compuesto de dos elementos principales: la sintaxis y la semántica. La *sintaxis* es el conjunto de operaciones mediante los cuales se construyen consultas. Por ejemplo, la sintaxis de SQL proporciona operaciones como `SELECT`, `FROM` y `JOIN`. La semántica de un lenguaje de consulta define cómo éstas son llevadas a cabo por el motor de consultas (p.ej. qué objetos son obtenidos y cómo son filtrados).

Morsa proporciona un lenguaje de consulta implementado con un *API de consulta* diseñada como una *interfaz fluida* (*fluent interface*) [17], esto es, un API Java tal que, cuando es instanciada, tiene forma de DSL embebido. La sintaxis del API está compuesta por una serie de clases que representan *condiciones* que deben ser satisfechas por los resultados de la consulta. Hay dos tipos principales de condiciones: *condiciones de atributo* y *condiciones de relación*, dependiendo de a qué sean aplicadas. Las condiciones de atributo pueden ser booleanas, numéricas o textuales. El API de consulta define clases como `BOOLEQ` (igualdad) y `BOOLNEQ` (desigualdad) para las condiciones booleanas, `LT` (menor que) y `GT` (mayor que) para las condiciones numéricas y `STREQ` (igualdad) y `STRLIKE` (similaridad) para las condiciones textuales. Las condiciones de relación (clase `REF`) permiten la navegación de relaciones, obteniendo todos los objetos accesibles a través de una relación que son de un tipo dado y que satisfacen una condición; estas condiciones permiten el anidamiento de otras condiciones de relación para así poder recorrer el modelo. Todas los tipos de condiciones soportan el método `AND`, el cual proporciona encadenamiento de condiciones en cortocircuito de forma similar al operador && de Java. A continuación se muestra un ejemplo de sintaxis para la consulta definida en la Sección 3:

```
Collection<EObject>result = new LinkedList<EObject>();
Collection<EObject>typeDeclarationList = morsaResource.loadObjects(typeDeclarationClass, true);
for (EObject typeDeclaration : typeDeclarationList) {
    Condition condition =
              new REF(methodDeclarationClass, "returnType",
                     new REF(simpleTypeClass, "name",
                            new STREQ(simpleNameClass,"fullyQualifiedName",
                                    typeDeclarationQName)))
              .AND(
                  new REF(methodDeclarationClass, "modifiers",
                         new BOOLEQ(modifierClass, "static", true))
                  .AND(
                      new REF(methodDeclarationClass, "modifiers",
                              new BOOLEQ(modifierClass, "public", true))
                  )
              );
    result.addAll(morsaResource.query(typeDeclaration, condition, -1, true));
}
```

Este código itera sobre todas las declaraciones de tipo (retornadas por el método `loadObjects`) y crea una condición para cada una de ellas que obtiene todas las declaraciones de método que tengan modificadores público y estático y cuyos tipos de retorno tengan el mismo nombre que el de la declaración de tipo.

El método `query` de `MorsaDriver` proporciona la semántica para las consultas. Este método transforma una condición en una consulta al almacenamiento persistente. Dicha consulta es representada en nuestro prototipo como una lista de `MorsaObject`s que se envían uno a uno a la base de datos en forma de documentos. Esto se hace así porque MongoDB no soporta la reunión de colecciones. Cada uno de estos documentos obtiene un conjunto de objetos que satisfacen una subcondición de la condición completa. El código listado anteriormente produce las siguientes consultas a la base de datos MongoDB:

1. Obtener todos los `SimpleClassName` tal que su *fullyQualifiedName* sea el mismo que el de la `TypeDeclaration` y que son descendientes de ésta.
2. Obtener todos los `SimpleType` tal que sus *simpleName* estén entre los obtenidos en el paso anterior.
3. Obtener todos los `MethodDeclaration` cuyos *returnType* estén entre los obtenidos en el paso anterior.
4. Obtener todos los `Modifier` con *static* con valor `true` que sean descendientes de los `MethodDeclaration` obtenidos en el paso anterior.
5. Obtener todos los `MethodDeclaration` que contengan los `Modifier`s obtenidos en el paso anterior.
6. Obtener todos los `Modifier` con *public* con valor `true` que sean descendientes de los `MethodDeclaration` obtenidos en el paso anterior.
7. Obtener todos los `MethodDeclaration` que contengan a los `Modifier`s obtenidos en el paso anterior.
8. Obtener todos los `TypeDeclaration` que contengan a los `MethodDeclaration` obtenidos en el paso anterior.

Como puede verse, la condición anidada de una condición `REF` es ejecutada en primer lugar y después usada para filtrar el subgrafo de objetos a recorrer. Esto permite la minimización del número de documentos a obtener de la base

de datos. Nótese que el método `AND` también minimiza dicho número al reducir el espacio de búsqueda a los objetos que sean descendientes de los obtenidos en pasos anteriores. El método `query` puede ser parametrizado para dejar de buscar a una cierta profundidad (-1 para profundidad ilimitada) y para no resolver los resultados de tal forma que se ahorre memoria, cargando bajo demanda cada uno de ellos cuando sean necesarios. Nótese también que el API de consultas hace uso de los metadatos definidos en la Sección 4.2 a la hora de filtrar los resultados de la condiciones desde las propias consultas a la base de datos (e.g, *Ancestors* para calcular descendencias).

7 Trabajo relacionado

La persistencia de modelos no es un campo de investigación novedoso; conforme el interés en MDE ha ido creciendo se han ido proponiendo muchos enfoques. La solución estándar de EMF es almacenar modelos en recursos (es decir, ficheros) XMI. Aunque hay otros enfoques tales como los ficheros indexados binarios [18], los repositorios de modelos son la solución más apropiada. Un *repositorio* es una solución de persistencia accesible remotamente por usuarios y herramientas y usualmente soportada por bases de datos, proporcionando características como transacciones y versionado. El repositorio de modelos EMF más maduro es *Connected Data Objects* (CDO) [7], ya que es el único capaz de manejar grandes modelos usando caga bajo demanda; es también el más usado. CDO se integra con EMF mediante su propia implementación de *Resource*, pero no es transparente porque necesita de preprocesado de modelos, como por ejemplo la creación de un modelo generador de Ecore para cada metamodelo y clases Java propias para soportar la carga bajo demanda. MongoEMF [19] es un repositorio de modelos EMF basado en MongoDB que proporciona integración transparente y consultas sencillas. Sin embargo, sólo aborda la escalabilidad en el cliente a través del particionado de modelos usando referencias entre recursos EMF, un mecanismo diseñado para proporcionar carga bajo demanda sencilla usando proxies; Morsa puede emular esto utilizando referencias entre `MorsaCollection`s. Por otro lado, Morsa es capaz de manejar modelos más grandes de forma más eficiente que CDO [10].

El enfoque estándar para consultas de modelos en EMF es EMFQuery [20], un API que proporciona métodos similares a SQL. Sin embargo, la evaluación de las consultas consiste en recorrer todo el modelo, lo cual es muy ineficiente. CDO proporciona soporte para consultas SQL y OCL, pero de forma bastante tosca ya que en el caso de SQL es obligatorio para formular una consulta conocer la estructura de la base de datos (que no es trivial) y en el caso de OCL el modelo es recorrido de forma completa como en EMFQuery.

8 Conclusiones y trabajo futuro

En este artículo hemos presentado Morsa, un repositorio de modelos enfocado a la escalabilidad de las aplicaciones cliente que acceden a modelos grandes. Morsa utiliza carga guardado incremental, carga bajo demanda y un API de consultas

para almacenar, cargar y consultar grandes modelos, respectivamente, sin sobrecargar la memoria de la aplicación cliente. El almacenamiento es llevado a cabo por una base de datos de documentos, lo que es novedoso, ya que los repositorios de modelos suelen trabajar con mappings objeto-relacionales. Nuestro trabajo futuro consiste en optimizar Morsa e implementar nuevas características, como el soporte para *traducir lenguajes de consulta* como OCL a nuestra API, *recolectar y analizar metadatos* sobre la estructura de los (meta)modelos para hacer más adaptativos los algoritmos de carga bajo demanda y considerar el *control de versiones* y la *sincronización* para dar soporte al desarrollo en equipo.

References

1. Monaheghehi et al.: MDE Adoption in Industry: Challenges and Success Criteria. Proceedings on the Workshop on Challenges in MDE, MoDELS september 2008, Toulouse (France), pages 54-59, Springer.
2. John Hutchinson, Jon Whittle, Mark Rouncefield, Steinar Kristoffersen: Empirical assessment of MDE in industry. ICSE 2011: 471-480
3. Kolovos, D., Paige, R., Polack, F.: Scalability: The Holy Grail of Model-Driven Engineering. Proceedings on the Workshop on Challenges in MDE, MoDELS september 2008, Toulouse (France), pages 35-47, Springer.
4. Selic, B.: Personal Reflections on Automation, Programming, Culture and Model-based Software Engineering. Automated Software Engineering vol. 15, number 3-4, pages 379-391, 2008, Springer.
5. The XML Metadata Interchange: http://www.omg.org/spec/XMI/.
6. The Eclipse Modeling Framework: http://www.eclipse.org/emf.
7. The CDO Model Repository: http://www.eclipse.org/cdo.
8. Prototipo de Morsa para EMF y MongoDB: http://www.modelum.es/morsa.
9. MongoDB: http://www.mongodb.org.
10. Espinazo-Pagán, J., Sánchez Cuadrado, J., García Molina, J.: Morsa: A Scalable Approach for Persisting and Accessing Large Models. Proceedings on the 14th International Model Driven Engineering Languages and Systems (MoDELS) Conference, Wellington (New Zealand), pages 77-92. 2011. Springer.
11. Stonebraker, M.: SQL Databases vs NoSQL Databases. Communications of the ACM, vol. 53, issue 4, pages 10-11, 2010. ACM.
12. Strauch, C.: NoSQL Databases. Stuttgart Media University, 2011. http://www.christof-strauch.de/nosqldbs.pdf.
13. DeCandia, G. et al.: Dynamo: Amazon's Higly-Available Key-value Store. Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles, pages 205-220, 2007. ACM.
14. Chang, F. et al.: Bigtable: A Distributed Storage System for Structured Data. 2006.
15. Grabats 2009 5th International Workshop on Graph-Based Tools: a reverse engineering case study, july 2009, Zurich (Switzerland) http://is.tm.tue.nl/staff/pvgorp/events/grabats2009/.
16. Binary JSON: http://www.bsonspec.org.
17. Fowler, M.: Domain-Specific Languages. 2010, Addison Wesley.
18. Jouault, F., Sottet, J.:An AmmA/ATL Solution for the Grabats 2009 Reverse Engineering Case Study. Grabats 2009 5th International Workshop on Graph-Based Tools, july 2009, Zurich (Switzerland).
19. MongoEMF: http://bryanhunt.wordpress.com/2011/03/15/mongo-emf/.
20. EMFQuery: http://www.eclipse.org/projects/project.php?id=modeling.emf.query.

A Family of Case Studies on Business Process Mining

Ricardo Pérez-Castillo, José A. Cruz-Lemus, Ignacio García-Rodríguez de Guzmán, and Mario Piattini

Alarcos Research Group, University of Castilla-La Mancha
Paseo de la Universidad, nº4 13071 – Ciudad Real (Spain)
{ricardo.pdelcastillo, joseantonio.cruz, ignacio.grodriguez, mario.piattini}@uclm.es

Abstract. Business processes, most of which are automated by information systems, have become a key asset in organizations. Unfortunately, uncontrolled maintenance implies that information systems age over time until they need to be modernized. During software modernization, ageing systems cannot be entirely discarded because they gradually embed meaningful business knowledge, which is not present in any other artifact. This paper presents a technique for recovering business processes from legacy systems in order to preserve that knowledge. The technique statically analyzes source code and generates a code model, which is later transformed by pattern matching into a business process model. This technique has been validated over a two year period in several industrial modernization projects. This paper reports the results of a family of case studies that were performed to empirically validate the technique using analysis and meta-analysis techniques. The study demonstrates the effectiveness and efficiency of the technique.

Keywords. Business Process, Static Analysis, Case Study, Meta-Analysis.

1 Motivation

Business processes are increasingly becoming an essential asset for organizations since they create value for customers and reflect all operations of an organization. Organizations adopt business process management through their enterprise information systems. Unfortunately, during software maintenance, the organizations' business processes do not reflect all changes that have occurred in legacy systems. Thereby, legacy information systems cannot be entirely discarded during its modernization since it might contain a considerable amount of latent meaningful business knowledge, which is not present anywhere else. As a result, all embedded business processes must be explicitly recovered in order to preserve this meaningful asset in the modernized information systems. The evolved system will thus support the current business processes and will also improve the ROI (*Return Of Investment*) of the legacy system, since it extends its lifespan.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 535-536, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

2 Proposal

This paper proposes MARBLE[1] (*Modernization Approach for Recovering Business processes from Legacy Systems*) is a framework that facilitates business process recovery through a path of three model transformations between four different abstraction levels. The first transformation takes legacy source code in the real world (level 0) and obtains a set of platform-specific models at level 1. The second transformation integrates every code model into a platform-independent KDM (Knowledge Discovery Metamodel) repository at level 2. Finally, the last transformation applies a pattern matching technique to detect business patterns in KDM models and builds business process models at level 3.

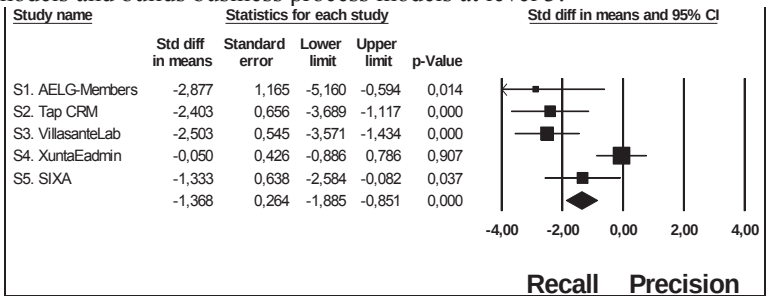


Fig. 1. Meta-Analysis results for recall and precision obtained by applying MARBLE

3 Empirical Validation Results

This work presents a family of case studies carried out over the last two years to validate the effectiveness and efficiency of MARBLE. The family was performed in five different industrial information systems: (i) a system managing a Spanish author organization; (ii) an open source CRM (Customer Relationship Management) system; (iii) an enterprise information system of the water and waste industry; (iv) an e-government system used in a Spanish local e-administration; (v) a high school LMS (Learning Management System). The study evaluates the effectiveness by means of the *precision* and *recall*, which measure the similarity between a mined business process *M* and a reference business process *R*. Precision indicates what proportion of *M* matches *R* (i.e., how exact *M* is), while recall indicates what proportion of *R* is present in *M* (i.e., how complete *M* is). The efficiency was assessed by considering the transformation time. A part from statistical methods, this study uses meta-analysis to quantify the difference in means for the recall and precision measures obtained in each particular system under study. The result obtained shows that the technique is suitable to recover business processes in an effective and efficient manner. However, according to the effectiveness, the recall values were better than the precision values. We believe that these results were obtained because much of the work was, in several cases, basically recovered from technical code.

[1] Pérez-Castillo, R., J. A. Cruz-Lemus, I. García-Rodríguez de Guzmán and M. Piattini (2012). "A Family of Case Studies on Business Process Mining using MARBLE." Journal of Systems and Software 86(6): 1370–1385.

Evolución de Sistemas Auto-Adaptables mediante Modelos en Tiempo de Ejecución*

María Gómez, Joan Fons, and Vicente Pelechano

Centro de Investigación en Métodos de Producción de Software
Universitat Politècnica de València
Camino de Vera s/n, E-46022, Spain
{magomez,jjfons,pele}@dsic.upv.es

Resumen La auto-adaptación se está convirtiendo en un requisito indispensable en los sistemas software. Los sistemas auto-adaptables son capaces de adaptar dinámicamente su comportamiento y estructura en ejecución en respuesta a variaciones en el sistema y el entorno. Los enfoques actuales de auto-adaptación tienden a asumir un mundo cerrado, en el que todos los fenómenos de interés son previstos en tiempo de diseño. Sin embargo, algunos comportamientos adaptativos no se pueden prever a priori. En este ámbito, se requieren técnicas que den soporte a la evolución en tiempo de ejecución de sistemas auto-adaptables. Este artículo propone una aproximación dirigida por modelos para desarrollar y evolucionar de forma sistemática sistemas auto-adaptables mientras están en ejecución. Específicamente, la aproximación utiliza *Modelos en Tiempo de Ejecución* para incorporar nuevas capacidades que no fueron previstas en el diseño inicial del sistema.

Keywords: Modelos en Tiempo de Ejecución; Evolución; Adaptación Dinámica

1. Introducción

Existe una creciente demanda de sistemas software que sean capaces de adaptarse, sin intervención humana, a cambios en las necesidades de los usuarios, variaciones en el entorno de ejecución, fallos y falta de disponibilidad de partes del sistema. La auto-adaptación se ha convertido en una de las principales aproximaciones para hacer frente a estas situaciones. La auto-adaptación proporciona a un sistema software la capacidad de adaptar su comportamiento y/o estructura en ejecución en respuesta a cambios en el entorno y en el propio sistema [10].

La *Ingeniería Dirigida por Modelos* (MDE) se ha centrado principalmente en la utilización de modelos durante el proceso de desarrollo software. Los modelos producidos en un proceso MDE también pueden jugar un papel importante en tiempo de ejecución, esta iniciativa se conoce como *Modelos en Tiempo de Ejecución* (*Models@run.time* [3]). Los modelos no sólo constituyen artefactos de primer orden en el proceso de desarrollo, también permiten monitorizar, validar y adaptar el comportamiento de un sistema

* This work has been supported by Conselleria d'Educación of Generalitat Valenciana under the program VALi+d and by Ministerio de Ciencia e Innovación (MICINN) under the project EVERYWARE TIN2010-18011.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 537-550, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

en ejecución. Por ejemplo, los modelos pueden describir los requisitos necesarios para reemplazar un componente del sistema que falle por otro, posibilitando que esto pueda llevarse a cabo de manera autónoma. Extender la aplicabilidad de los modelos de diseño al entorno de ejecución introduce beneficios significativos. Una ventaja clave consiste en que los modelos pueden proporcionar una base semántica para la toma de decisiones autónomas en ejecución. Durante los últimos años se han utilizado de forma exitosa técnicas y herramientas MDE para adaptar sistemas en ejecución en respuesta a cambios en el contexto (ej. [13],[16]).

Una de las principales características de los sistemas auto-adaptables es que son conscientes del entorno de ejecución y pueden adaptarse en respuesta a cambios sin intervención humana. En la práctica, los ingenieros identifican en tiempo de diseño los cambios que pueden aparecer en el entorno y en el sistema, y especifican cómo el sistema debe comportarse para afrontar dichos cambios. Sin embargo, en entornos altamente dinámicos, todas las eventualidades no se pueden prever a priori. Por ejemplo, nuevos usuarios, dispositivos o software, que no se han considerado en tiempo de diseño, pueden ser añadidos sin previo aviso, o una política de comportamiento del sistema puede aparecer o cambiar. La ingeniería de sistemas auto-adaptables ha experimentado un progreso significativo en la última década ([17], [4], [10]). Sin embargo, los procesos de evolución y mantenimiento han recibido escaso interés hasta ahora. Las aproximaciones actuales de auto-adaptación asumen un mundo cerrado, donde todos los fenómenos de interés han sido predefinidos. Los sistemas auto-adaptables deben estar abiertos a evolución con el objetivo de manejar nuevos requisitos y no quedar obsoletos.

Típicamente, las actividades de evolución y mantenimiento del software se han llevado a cabo con el sistema parado [11]. La evolución del sistema se consigue ejecutando secuencias de comandos escritas a nivel del lenguaje de programación. Estas soluciones suelen ser complejas y propensas a errores, además dificultan la validación y el mantenimiento. En sistemas auto-adaptables, los procesos de evolución y mantenimiento tienen que ser llevados a cabo mientras el sistema está en ejecución. Para abordar las tareas de evolución, proponemos la utilización de *Modelos en Tiempo de Ejecución* donde los modelos capturan en todo momento las descripciones del sistema y su entorno necesarias para entender su situación actual, y las posibles necesidades de auto-adaptación. La evolución se realiza a nivel de modelo (en vez de a nivel de implementación) permitiendo razonar sobre el sistema sin atender a detalles técnicos. La utilización de modelos proporciona independencia tecnológica, por tanto se puede definir un proceso de evolución genérico que se puede aplicar en diferentes sistemas. Por otro lado, los cambios de evolución son planificados y aplicados sobre los modelos primero, los cuales ilustran el estado del sistema como resultado de la evolución. La nueva versión del sistema puede ser analizada para determinar si se cumplen ciertas propiedades (por ejemplo, se asegura que no se violan ciertas restricciones), antes de reflejar los cambios en el sistema en funcionamiento.

En este artículo se propone en primer lugar aplicar la filosofía MDE a los sistemas auto-adaptables. Se definen *Lenguajes Específicos de Dominio (DSL)* para formalizar los conceptos involucrados en el desarrollo de sistemas auto-adaptables, modelos y herramientas de soporte. En segundo lugar, se propone una aproximación basada en modelos para evolucionar de forma sistemática sistemas auto-adaptables en ejecución.

La evolución implica la mejora, corrección e incorporación de nuevas capacidades en un sistema auto-adaptable mientras está en operación. Hemos implementado un motor de evolución denominado *MovE (**Mo**del-based **E**volution **E**ngine)* que se encarga de automatizar el proceso de evolución. MovE utiliza técnicas de *Modelos en Tiempo de Ejecución* y *Computación Autónoma*, para manejar la complejidad inherente a los procesos de evolución de software. En un trabajo previo desarrollamos un motor de reconfiguración basado en modelos *MoRE (Model-based Reconfiguration Engine)* [5] para dar soporte a la adaptación dinámica de sistemas en respuesta a cambios en el entorno de ejecución. El motor MovE se integra con MoRE para complementar las capacidades de adaptación con capacidades de evolución.

El resto del artículo se organiza de la siguiente forma. La Sección 2 presenta un caso de estudio para ilustrar la aplicabilidad de la propuesta. La Sección 3 describe los lenguajes y modelos definidos para desarrollar sistemas auto-adaptables. La Sección 4 presenta la aproximación propuesta para evolucionar sistemas auto-adaptables en ejecución. Finalmente, la Sección 5 y la Sección 6 presentan el trabajo relacionado y las conclusiones respectivamente.

2. Caso de Estudio

En esta sección se presenta un caso de estudio para ilustrar la aplicabilidad de la propuesta. El caso de estudio es acerca de una aplicación móvil adaptativa que sirve para asistir a los usuarios en visitas a la Universidad Politécnica de Valencia (UPV). La aplicación llamada *UPVAssistant*, proporciona un conjunto de servicios que permiten: obtener itinerarios para llegar a distintos edificios o lugares del campus dependiendo de la localización del usuario, acceder a la biblioteca digital, acceder al servicio de reservas deportivas, obtener las instalaciones (ej. cafeterías, bibliotecas) más cercanas a la ubicación del usuario, etc. En función del contexto de ejecución la aplicación es capaz de reconfigurar sus servicios de forma autónoma. A continuación se presentan dos escenarios de adaptación y evolución que pueden darse en el sistema:
Escenario de adaptación. Un usuario llega a la UPV y utiliza la aplicación *UPVAssistant* para obtener un itinerario que le guíe al departamento de Sistemas Informáticos donde tiene una reunión. La aplicación utiliza un servicio que le indica el itinerario a seguir desde su ubicación actual hasta el destino. En su camino al departamento, la conexión con el servicio de itinerario se pierde. Entonces el servicio de itinerario se reemplaza automáticamente por un servicio de mapas que ofrece la misma funcionalidad y puede seguir guiando al usuario.
Escenario de evolución. Cuando la aplicación *UPVAssistant* está en ejecución, el diseñador observa que el servicio de mapas consume muchos recursos y decide insertar nuevo comportamiento adaptativo para mejorar la calidad de la aplicación. El diseñador inserta una nueva política de adaptación en el sistema en ejecución, indicando que cuando la batería del dispositivo móvil es menor al veinte por cien, el servicio de mapas se reemplaza automáticamente por el servicio de itinerario que proporciona la misma funcionalidad.

A partir de los escenarios anteriores, se puede observar que algunas adaptaciones pueden ser anticipadas en tiempo de diseño y preparar el sistema para responder de

forma autónoma. Sin embargo, cuando la aplicación está en ejecución pueden aparecer nuevos requisitos que requieren la evolución del sistema para dar soporte a nuevas adaptaciones no previstas en el diseño original.

3. Desarrollo de Sistemas Auto-Adaptables

Los sistemas auto-adaptables pueden razonar acerca de su estado interno y del entorno. Una técnica utilizada para conseguir sistemas auto-adaptables es la introducción de un bucle de control en el sistema que se encarga de guiar el comportamiento autónomo [4]. IBM propone un modelo de referencia para bucles de control, conocido como MAPE-K [7]. De acuerdo con esta estrategia, el sistema monitoriza continuamente el entorno de ejecución y su estado interno, analiza la información, planifica cómo adaptar el sistema, y ejecuta un plan de adaptación basado en el conocimiento del sistema y el entorno. Este conocimiento puede representarse de diferentes formas, como por ejemplo, descripciones textuales, metadatos en código, o modelos. En la aproximación MovE se propone el uso de abstracciones basadas en modelos. El conocimiento contiene descripciones sobre el sistema, las políticas de adaptación y el entorno. En MovE se definen Lenguajes Específicos de Dominio y herramientas para construir los modelos que constituyen el conocimiento del sistema. Cuando el sistema auto-adaptable está en ejecución, el diseñador debe poder evolucionar el conocimiento del sistema para incorporar nuevas capacidades que permitan responder a requisitos no previstos en el diseño original. Estos modelos evolucionados serán validados y finalmente procesados para calcular los cambios reales a aplicar sobre los sistemas en tiempo de ejecución.

3.1. Modelado del Sistema

Para razonar sobre los cambios necesarios de adaptación y evolución en el sistema, se necesita una representación actualizada de los componentes del sistema en funcionamiento. Se ha utilizado un modelo SoaML (Service oriented architecture Modeling Language)[1] (ver Figura 1b) para representar la arquitectura del caso de estudio descrito en este trabajo. SoaML es el estándar propuesto por el Object Management Group (OMG) para modelado de arquitecturas orientadas a servicios. Además de la representación de la arquitectura, los sistemas auto-adaptables necesitan un mecanismo de descripción de variabilidad para definir las posibles configuraciones por las que puede pasar el sistema. El enfoque de *Líneas de Producto Software (SPL)* y concretamente las *Líneas de Producto Software Dinámicas (DSPL)*[6] permite representar un sistema auto-adaptable como una familia de configuraciones de un mismo sistema con un mecanismo de adaptación para pasar de una configuración a otra. En *UPVAssistant* se utiliza un *Modelo de Características* (ver Figura 1a) para definir de forma intensional la configuración del sistema y las posibles variantes. Las características verdes, representan características activas, mientras que las características rojas representan variantes que podrán ser activadas en el futuro. Cada configuración del sistema se representa mediante un conjunto de características activas. Se ha definido un *Metamodelo de Configuraciones* (Figura 2) para representar las configuraciones del sistema a partir de un modelo

[1] http://www.omg.org/spec/SoaML/

de características. El elemento raíz del metamodelo es la clase *ConfigurationModel*. El *ConfigurationModel* tiene una referencia al modelo de características (*FeatureModel*) y está compuesto por un conjunto de elementos *FeatureState*. Cada *FeatureState* tiene una referencia a una característica (*Feature*) del modelo de características y un atributo estado (*state*) que representa el estado de la característica. Hay dos tipos posibles de estados: Activa e Inactiva (*Active e Inactive*).

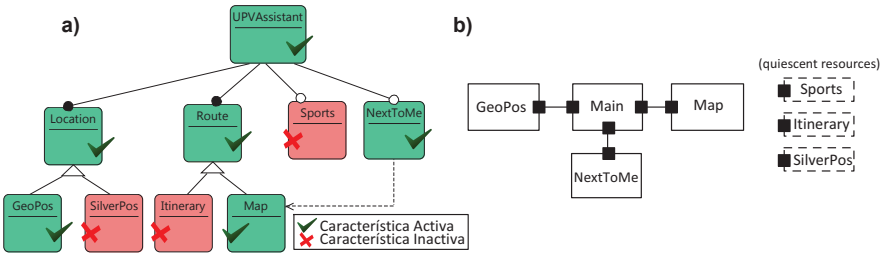


Figura 1. a) Modelo de Características. b) Modelo de Arquitectura SoaML

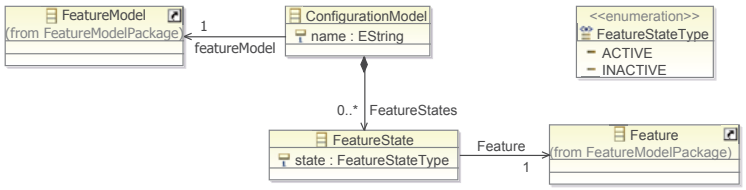


Figura 2. Metamodelo de Configuraciones.

El *Modelo de Características* representa la funcionalidad del sistema de forma taxonómica, pero las características de un modelo de características son simplemente símbolos. Relacionando características con otros modelos, como especificaciones de comportamiento o datos, las dota de semántica. Utilizamos un *Modelo de Weaving* para especificar la correspondencia entre cada característica del *Modelo de Características* y los elementos del *Modelo de Arquitectura*. Un *Modelo de Weaving* es un tipo especial de modelo utilizado para establecer y manejar relaciones entre los elementos de varios modelos. Este tipo de modelos es habitual en estrategias de desarrollo MDE y permite mantener los modelos de origen y destino sin modificaciones.

3.2. Modelado de las Políticas de Adaptación

Los sistemas auto-adaptables necesitan un mecanismo para representar las políticas de adaptación que indican cuándo y cómo cambiar de una configuración a otra. En este caso de estudio se utiliza un *Modelo de Resoluciones* para representar las políticas de

adaptación. El *Modelo de Resoluciones* se define a partir del *Modelo de Características*. Una *Resolución* está relacionada con una condición de contexto y define los cambios, en términos de activación/desactivación de características, a realizar en el sistema cuando la condición se cumple. Por ejemplo, cuando el servicio *GeoPos* deja de estar disponible se reemplaza por el servicio *SilverPos*. La siguiente resolución define este escenario: $R_{GeoPos-Un} = \{(GeoPos, Desactivar), (SilverPos, Activar)\}$.

Para formalizar las políticas de adaptación, se ha definido un *Metamodelo de Resoluciones* (ver Figura 3a). El elemento raíz del metamodelo es la clase *ResolutionModel*. El *ResolutionModel* está compuesto por un conjunto de elementos *Resolution* que representan las diferentes resoluciones. Cada *Resolution* está compuesta por una condición (*Condition*), un conjunto de acciones (*Action*) y una guarda (*Guard*). Una acción tiene una referencia a una característica del Modelo de Características y un tipo de acción que puede ser *Activar* o *Desactivar*. La Figura 3b ilustra el *Modelo de Resoluciones* de *UPVAssistant*.

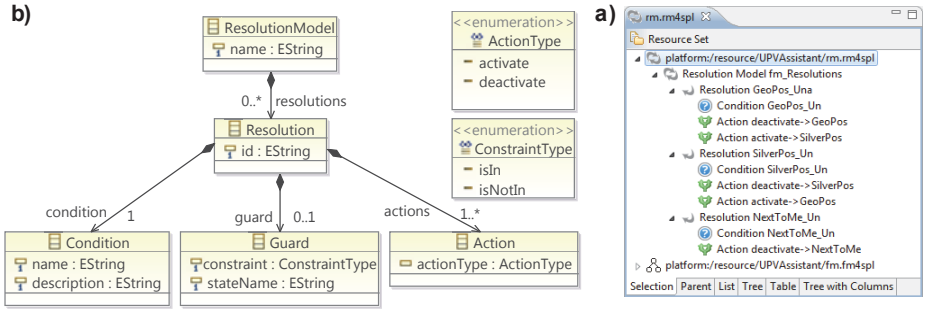


Figura 3. a) Metamodelo de Resoluciones. b) Modelo de Resoluciones de UPVAssistant

Además el sistema necesita una representación para razonar sobre el entorno de ejecución. En el caso de estudio se utiliza un *Modelo de Contexto* que se actualiza periódicamente con la información observada del contexto durante la ejecución del sistema. Concretamente, en *UPVAssistant* se ha usado un *Modelo de Contexto* basado en una ontología OWL[2]. Las condiciones asociadas a las resoluciones (del *Modelo de Resoluciones*) se definen sobre el *Modelo de Contexto* como expresiones booleanas. El *Modelo de Resoluciones* se utiliza en ejecución para iniciar las acciones de adaptación. Cuando se cumple la condición asociada a una resolución, la resolución se dispara provocando la adaptación del sistema.

3.3. Herramientas de Soporte

Esta sección describe las herramientas de soporte para la creación de cada uno de los modelos del sistema. Se ha desarrollado una herramienta denominada Moskitt4SPL[3]

[2] http://www.w3.org/TR/owl-guide/
[3] http://www.m4spl.org/

que integra editores para la creación de *Modelos de Características*, *Modelos de Configuraciones* y *Modelos de Resoluciones*. Para la creación de *Modelos de Características* y *Modelos de Configuraciones* se han desarrollado dos editores gráficos con *Eclipse Graphical Modeling Framework (GMF)*[4]. Para la creación de *Modelos de Resoluciones* se ha generado un editor en árbol con *Eclipse Modeling Framework (EMF)*[5]. El *Modelo de Arquitectura* conforma al metamodelo de SoaML y ha sido creado usando las capacidades de EMF. El *Modelo de Características* y el *Modelo de Arquitectura* se toman como entrada en la herramienta ATLAS Model Weaver[6] para crear el *Modelo de Weaving*. Finalmente, el *Modelo de Contexto* ha sido creado con la herramienta Protégé-OWL[7]. La Figura 4 resume los modelos creados y las herramientas de soporte[8].

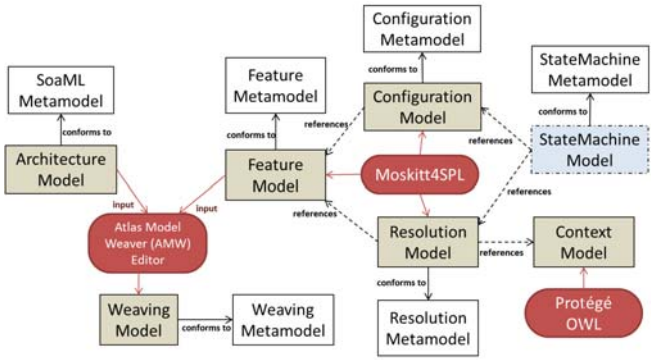


Figura 4. Modelos, metamodelos y herramientas de soporte.

4. MovE: Soporte a la Evolución de Sistemas Auto-adaptables

Hasta ahora el trabajo ha presentado cómo especificar el conocimiento de un sistema auto-adaptable mediante modelos. Estos modelos se consultan y actualizan en tiempo de ejecución para adaptar dinámicamente un sistema en ejecución. Con el paso del tiempo, todo sistema software requiere evolución y mantenimiento [9]. El diseñador debe poder evolucionar los modelos del conocimiento del sistema para incorporar nuevas capacidades. Esta sección presenta MovE, una aproximación metodológica basada en modelos para automatizar la evolución de sistemas auto-adaptables. El objetivo de MovE es insertar nuevas capacidades (no anticipadas en tiempo de diseño) en un sistema auto-adaptable en ejecución, mediante la inclusión de otro bucle de control para

[4] www.eclipse.org/gmf
[5] www.eclipse.org/emf
[6] http://www.eclipse.org/gmt/amw/
[7] http://protege.stanford.edu/overview/protege-owl
[8] El Modelo de Máquina de Estados que aparece en la figura se presenta en una sección posterior

automatizar la evolución. El proceso de evolución en MovE comprende 4 fases basadas en las fases del bucle MAPE-K: *Monitorizar, Analizar, Planificar y Ejecutar*. Esta estrategia junto con la utilización de modelos en tiempo de ejecución permite desacoplar el proceso de evolución del sistema en ejecución. Los cambios de evolución se planifican y analizan sobre modelos (a mayor nivel de abstracción) antes de hacerlos efectivos en el sistema real en funcionamiento. Así, el proceso de evolución es genérico e independiente de la tecnología de implementación.

4.1. Monitorizar

En la primera fase, un componente *Monitor* se encarga de detectar la aparición de nuevos requisitos e iniciar el proceso de evolución. El proceso de evolución se inicia de dos formas: (1) *reactiva*, el ingeniero decide iniciar y guiar la evolución para mejorar, modificar o refinar el sistema; y (2) *proactiva*, el sistema inicia la evolución de forma autónoma. El *Monitor* cuenta con un conjunto de sensores para monitorizar el sistema y detectar la aparición de eventos que provocan la evolución del sistema. Por ejemplo, se definen oyentes para detectar modificaciones sobre los modelos que describen el sistema (llevadas a cabo por el ingeniero). Estas modificaciones inician el proceso de evolución reactiva. Otros oyentes escuchan eventos de contexto para detectar la aparición de nuevos dispositivos, recursos, usuarios, etc. e inician el proceso de evolución proactiva. En el caso de estudio presentado en este trabajo se ilustra el proceso de evolución reactiva.

4.2. Analizar

El objetivo de la segunda fase es la obtención de una nueva versión de las descripciones (modelos) del sistema, donde se incorporen los cambios requeridos por la evolución. Es inadmisible evolucionar un sistema a una configuración inválida o inconsistente. Con el objetivo de no introducir errores ni inconsistencias en el sistema en ejecución, los cambios deben ser analizados antes de aplicarlos en el sistema. Para analizar la evolución, MovE implementa la estrategia *Replication with Validation* [8] que propone la creación y mantenimiento de copias de los recursos (a ser evolucionados) con fines de validación. Siguiendo este enfoque, los cambios se aplican y analizan en copias de los modelos primero. Si no se detectan errores, entonces la evolución puede ser aplicada en el sistema en funcionamiento de forma segura. Los pasos llevados a cabo en la fase de análisis son:

Paso 1. Extraer una copia del conocimiento. El primer paso de la fase de análisis es extraer una copia de los modelos en ejecución que constituyen el conocimiento del sistema (ej. modelo de características, modelo de resoluciones, etc.).

Paso 2. Evolucionar el conocimiento a una nueva versión. Esta fase proporciona mecanismos para asistir la evolución del conocimiento a una nueva versión. Por ejemplo, cuando la aplicación *UPVAssistant* está en ejecución, el diseñador observa que

el servicio de mapas (*Map*) consume muchos recursos y decide insertar nuevo comportamiento adaptativo para mejorar la calidad de la aplicación. El diseñador inserta una nueva política de adaptación indicando que cuando la batería del dispositivo móvil es menor al veinte por cien, se reemplace el servicio *Map* por el servicio de itinerario (*Itinerary*). El diseñador tiene que modificar el *Modelo de Resoluciones* insertando una nueva resolución. La nueva resolución se define de la siguiente forma: $R_{Bat<20}$={(Map,Desactivar),(Itinerary,Activar)}.

MovE facilita los editores gráficos de los modelos del sistema. Los editores muestran copias de los modelos en ejecución. EMF proporciona un framework de notificación (*EMF Notifier and Adapter*) que permite registrar oyentes que son notificados automáticamente cuando se modifica un modelo. Todos los modelos abiertos en los editores gráficos (en MovE) registran un oyente (*Adapter*) que se encarga de notificar cuando el modelo es modificado.

Paso 3. Validar la nueva versión evolucionada. Cuando se recibe una notificación de modificación de un modelo, los cambios asociados a cada evolución deben ser validados antes de reflejarlos en el sistema en funcionamiento. Para comprobar y garantizar que la evolución no produce errores ni deja al sistema en un estado inconsistente, se ha propuesto un proceso de validación. El proceso de validación comprueba si la nueva versión del sistema cumple una serie de invariantes. Para ello, se han predefinido invariantes que deben satisfacerse siempre en el sistema. El proceso de validación consiste en dos pasos: a) Obtención del espacio de adaptación del sistema, y b) Comprobación de invariantes y refinamiento de los modelos.

a) Obtención del espacio de adaptación.

La ejecución de un sistema adaptativo puede ser abstraída como una máquina de estados ([17,2]) donde cada estado representa una configuración distinta del mismo sistema y cada transición representa una reconfiguración entre configuraciones. Esta representación gráfica ofrece una visión global del proceso completo de reconfiguración y reduce el esfuerzo y la complejidad durante el análisis. A partir de la configuración actual del sistema (*Modelo de Configuración*) y el conjunto de reglas de adaptación (*Modelo de Resoluciones*), es posible obtener el espacio de adaptación del sistema en forma de máquina de estados. Usando ATL, hemos implementado una transformación que define relaciones, por medido de reglas modelo a modelo, para generar automáticamente un *Modelo de Máquina de Estados* (definido de acuerdo a un metamodelo de máquina de estados) a partir de un *Modelo de Configuración* y un *Modelo de Resoluciones*. La Figura 5 (izquierda) ilustra el espacio de adaptación de *UPVAssitant* obtenido a partir del *Modelo de Resoluciones inicial*. La Figura 5 (derecha) ilustra el espacio de adaptación resultante a partir del *Modelo de Resoluciones evolucionado*.

b) Comprobación de invariantes y refinamiento de los modelos.

Tras obtener el espacio de adaptación, el siguiente paso es seleccionar un invariante y comprobar si se satisface en la nueva versión de los modelos del sistema. Si el invariante se satisface, entonces la nueva versión es válida y continúa el proceso de evolución. Si el invariante no se cumple, se aplica un refinamiento a los modelos para garantizar que se cumpla.

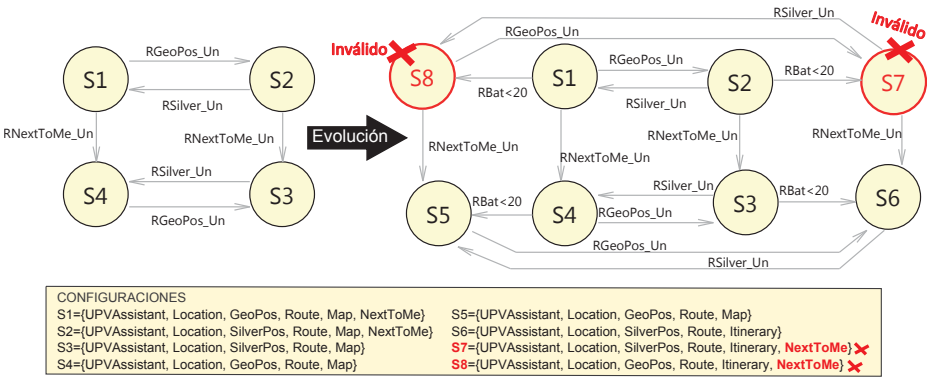


Figura 5. Espacio de adaptación inicial (izquierda) y espacio de adaptación evolucionado (derecha).

Uno de los invariantes definidos expresa que *después de una reconfiguración el sistema nunca alcanza una configuración inválida*. Se consideran configuraciones inválidas aquellas que violan alguna restricción de variabilidad del modelo de características. El refinamiento asociado a este invariante garantiza que todas las reconfiguraciones del sistema son seguras (es decir, no llevan a un estado inválido). A partir de la configuración inicial, el refinamiento iterativamente comprueba las reconfiguraciones que se pueden disparar en cada configuración del espacio de adaptación. Si alguna reconfiguración lleva a una configuración inválida, el refinamiento introduce guardas a las resoluciones que disparan esas reconfiguraciones inseguras, con el fin de evitar estados inconsistentes en el sistema. Por ejemplo, el *Modelo de Características* del caso de estudio (Figura 1a) contiene una restricción de variabilidad (de tipo *"requiere"*) entre el servicio *NextToMe* y el servicio *Map*. Esta restricción indica que el servicio *NextToMe* requiere al servicio *Map* para llevar a cabo su función. En el espacio de adaptación generado a partir de la nueva versión evolucionada (Figura 5 derecha), se puede observar que eventualmente cuando el sistema está en la configuración S1 o S2, si se dispara la nueva resolución $R_{Bat<20}$ el sistema se reconfigura a un estado inconsistente (S8 y S7 respectivamente). S8 y S7 son configuraciones inválidas porque está activa la característica *NextToMe* y está inactiva la característica *Map*. Con el objetivo de evitar configuraciones inconsistentes en ejecución, el refinamiento añade una guarda a la resolución $R_{Bat<20}$ para evitar que se dispare en las configuraciones S1 y S2. Se utilizan guardas en vez de eliminar directamente la resolución $R_{Bat<20}$ para no perder capacidad de reconfiguración puesto que existen otras configuraciones en el espacio de adaptación (S3, S4) donde disparar la resolución no genera comportamiento inválido. La resolución refinada quedaría de la siguiente forma: [!S1;!S2]$R_{Bat<20}$ = {(Map,Desactivar),(Itinerary,Activar)}.

Los refinamientos también han sido implementados como transformaciones ATL. El refinamiento toma como entrada un *Modelo de Máquina de Estados* y un *Modelo de Resoluciones* y genera un *Modelo de Máquina de Estados Refinado* y un *Modelo de Resoluciones Refinado*. El refinamiento añade guardas en la nueva resolución insertada

para evitar que se dispare en determinadas configuraciones. La Figura 6 resume los modelos y transformaciones implicados en el proceso de validación.

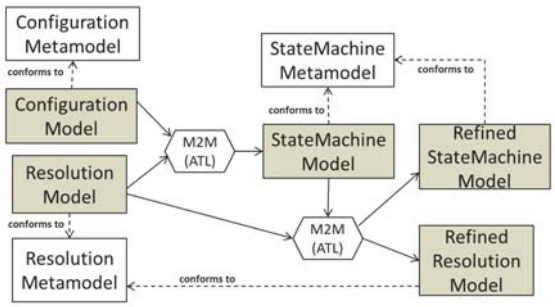


Figura 6. Transformaciones involucradas en el proceso de validación.

4.3. Planificar

El sistema en ejecución no se ha modificado todavía, los cambios se han aplicado en una copia de los modelos. Esta fase proporciona mecanismos para generar un *Plan de Evolución* que contiene las acciones necesarias para modificar el sistema en ejecución como resultado de la evolución ocurrida. En primer lugar se calculan las diferencias entre los modelos en ejecución y la nueva versión evolucionada. Para obtener las diferencias se utiliza la facilidad *EMF Compare*[9] que permite obtener las diferencias entre dos modelos EMF. Se obtienen conjuntos de *incrementos* (Δ) y *decrementos* (∇) de los modelos del conocimiento afectados por la evolución. Estos conjuntos ilustran las diferencias entre versiones en términos de elementos de modelos. La evolución del caso de estudio afecta al *Modelo de Resoluciones* y se obtienen los siguientes *incrementos* y *decrementos*:
Δ(ResolutionModel) = { [!S1;!S2]$R_{Bat<20}$ }
∇(ResolutionModel) = { }
Estas diferencias se utilizan para construir el *Plan de Evolución* encargado de actualizar los modelos usados por el sistema en ejecución. A partir de un metamodelo en ecore, EMF ofrece una API para manejar los modelos que conforman al metamodelo. La API proporciona la clase *Factory* que permite crear nuevas instancias de las clases definidas en el metamodelo, una interfaz Java y una implementación para cada una de las clases del metamodelo. Estas clases proporcionan métodos para acceder y cambiar la información de las instancias especificadas en el modelo. Hemos extendido estas clases para implementar las primitivas básicas de evolución que definen la adición, eliminación y modificación de elementos en cada uno de los modelos que constituyen el conocimiento del sistema (por ejemplo Añadir Resolución, Eliminar Resolución, etc.). El Δ(ResolutionModel) se traduce en el plan de evolución:

[9] http://www.eclipse.org/emf/compare/

PE={AddResolution([!S1;!S2]$R_{Bat<20}$}.
El conjunto de decrementos es vacío en este caso.

4.4. Ejecutar

Finalmente, esta fase es responsable de ejecutar el *Plan de Evolución* para evolucionar el sistema en ejecución a la nueva versión. La evolución puede implicar realizar cambios únicamente en los modelos en tiempo de ejecución (ej. modificar una condición de contexto, añadir una nueva política de adaptación) como en el ejemplo ilustrado en esta sección. Pero la evolución también puede implicar cambios más complejos que afecten a la arquitectura del sistema, donde además de actualizar los modelos se debe actualizar el sistema real (ej. adición de un nuevo servicio al sistema). En este caso, los cambios dependen de las tecnologías concretas de la plataforma destino en la que están implementados los sistemas. Mediante transformaciones modelo a texto se podría automatizar la generación de las implementaciones concretas de las acciones de evolución. El caso de estudio *UPVAssistant* se ha implementado en OSGi[10]. Para comprobar la viabilidad de la propuesta, se han definido las acciones de evolución en el sistema *UPVAssistant* de forma manual, como primer paso para abstraer los generadores de código que serán desarrollados en trabajos futuros.

5. Trabajo Relacionado

En esta sección se presentan aproximaciones que siguen un enfoque dirigido por modelos para desarrollar sistemas auto-adaptables y se comparan con la aproximación propuesta.

Morin et al. [12] proponen una combinación de ingeniería dirigida por modelos y modelado orientado a aspectos para dar soporte a la auto-adaptación. Los aspectos se componen dinámicamente para producir modelos de configuración, a partir de los cuales se generan los scripts necesarios para adaptar la arquitectura de un sistema en funcionamiento. Al igual que en nuestra propuesta, la variabilidad se gestiona de forma dinámica en ejecución y no es necesario enumerar todas las posibles configuraciones del sistema. Rouvoy et al. proponen MUSIC [16], un enfoque dirigido por modelos para facilitar el desarrollo y ejecución de aplicaciones móviles que son capaces de auto-adaptarse de acuerdo a variaciones en la calidad del servicio en entornos ubicuos. A diferencia de nuestro trabajo, los modelos se fijan en tiempo de diseño y no se proporcionan mecanismos para modificarlos en tiempo de ejecución. Parra et al. [15] proponen CAPucine, una línea de productos dinámica para desarrollar sistemas adaptativos basados en servicios. Utilizan un modelo de características para gestionar la variabilidad. Cada característica del modelo de características se corresponde con un modelo parcial del producto. Estos modelos se componen y transforman para generar el producto final. Las aproximaciones mencionadas anteriormente siguen un enfoque dirigido por modelos igual que MovE. Sin embargo, ninguna de ellas aborda mecanismos para evolucionar sistemáticamente los modelos del sistema, con el objetivo de hacer frente a situaciones imprevistas y nuevos requisitos.

[10] http://www.osgi.org/

Existen algunas propuestas que posibilitan la evolución manual en ejecución de sistemas adaptativos. Por ejemplo, Andrade et al. [1] definen una aproximación para evolucionar la lógica de adaptación del sistema. La lógica de adaptación se representa mediante reglas acción-condición que especifican las acciones a realizar en respuesta a cambios en el contexto. Esta reglas pueden ser cambiadas en tiempo de ejecución. Nuestra aproximación, permite evolucionar las reglas de adaptación y también el resto de modelos del sistema. Morin et al. [14] proponen una plataforma para unificar los eventos de evolución de diseño y los eventos de adaptación del sistema. Cuando los modelos de diseño son evolucionados, se analizan eventos del sistema en ejecución para determinar si es adecuado realizar la evolución en el sistema. Al igual que en MovE, la evolución del sistema se lleva a cabo a través de la evolución de los modelos de diseño conectados con el sistema. MovE proporciona editores gráficos para modificar cada uno de los modelos, y ofrece refinamientos para evitar errores de forma automática antes de reflejar los cambios en el sistema en funcionamiento.

6. Conclusiones y Trabajo Futuro

Las aproximaciones actuales de desarrollo de sistemas auto-adaptables tienden a asumir un mundo cerrado donde todos los posibles comportamientos han sido predefinidos en tiempo de diseño. Sin embargo, con el paso del tiempo, el diseñador debe evolucionar el conocimiento del sistema para incorporar nuevas capacidades. En primer lugar, hemos definido DSLs y herramientas para modelar sistemas auto-adaptables. Estos modelos pueden ser reutilizados en ejecución para adaptar dinámicamente el sistema en ejecución. A continuación se presenta MovE, una aproximación basada en modelos para automatizar la evolución de sistemas auto-adaptables en ejecución. Los cambios requeridos por la evolución se aplican en una copia de los modelos inicialmente. La nueva versión evolucionada de los modelos se analiza antes de reflejarla en el sistema en funcionamiento. Si se detecta algún problema, los modelos se refinan mediante transformaciones modelo-a-modelo para asegurar que la evolución es válida. Finalmente, se calcula un plan de evolución y se ejecuta en el sistema. La utilización de modelos en tiempo de ejecución permite definir un proceso de evolución independiente de plataforma que puede ser aplicado en diferentes sistemas.

Como trabajo futuro, planeamos implementar las herramientas para dar soporte a la última fase del proceso de evolución (fase *Ejecutar*). Concretamente, generadores de modelo a código para obtener de forma automática las acciones de evolución en distintas implementaciones. También, se pretende integrar el analizador de modelos de características *FaMa-FW*[11] en el proceso de validación para analizar las nuevas configuraciones generadas como resultado de la evolución. Finalmente, se planea aplicar y seguir validando la propuesta en distintos ámbitos de aplicación.

Referencias

1. Andrade, S.S., Macedo, R.J.d.A.: A non-intrusive component-based approach for deploying unanticipated self-management behaviour. In: Proc. of the ICSE Workshop on SEAMS. pp.

[11] http://www.isa.us.es/fama/

152–161 (2009)
2. Bencomo, N., Grace, P., Flores, C., Hughes, D., Blair, G.: Genie: supporting the model driven development of reflective, component-based adaptive systems. In: Proc. of ICSE '08. pp. 811–814. ACM, New York, NY, USA (2008)
3. Blair, G., Bencomo, N., France, R.B.: Models@run.time. Comp. 42, 22–27 (2009)
4. Brun, Y., Marzo Serugendo, G., Gacek, C., Giese, H., Kienle, H., Litoiu, M., Müller, H., Pezzè, M., Shaw, M.: Software engineering for self-adaptive systems. chap. Engineering Self-Adaptive Systems through Feedback Loops, pp. 48–70. Springer-Verlag, Berlin, Heidelberg (2009)
5. Cetina, C., Giner, P., Fons, J., Pelechano, V.: Autonomic computing through reuse of variability models at runtime: The case of smart homes. Comp. 42, 37–43 (2009)
6. Hallsteinsen, S., Hinchey, M., Park, S., Schmid, K.: Dynamic Software Product Lines. Computer 41(4), 93–95 (2008)
7. IBM: An architectural blueprint for autonomic computing. (2003)
8. King, T., Ramirez, A., Cruz, R., Clarke, P.: An integrated self-testing framework for autonomic computing systems. Journal of Computers 2(9) (2007)
9. Lehman, M.M.: Software's future: Managing evolution. IEEE Sof 15, 40–44 (1998)
10. de Lemos, R., Giese, H., Müller, H., Shaw, M.: Software Engineering for Self-Adpaptive Systems: A second Research Roadmap. In: Software Engineering for Self-Adaptive Systems. No. 10431 in Dagstuhl Seminar Proceedings (2011)
11. Mens, T., Demeyer, S., Mens, T.: Introduction and roadmap: History and challenges of software evolution. In: Software Evolution, pp. 1–11 (2008)
12. Morin, B., Barais, O., Nain, G., Jezequel, J.M.: Taming dynamically adaptive systems using models and aspects. In: Proc. of ICSE '09. pp. 122–132 (May 2009)
13. Morin, B., Barais, O., Jezequel, J.M., Fleurey, F., Solberg, A.: Models@ Run.time to Support Dynamic Adaptation. Computer 42, 44–51 (2009)
14. Morin, B., Ledoux, T., Hassine, M.B., Chauvel, F., Barais, O., Jezequel, J.M.: Unifying runtime adaptation and design evolution. In: Proc. of CIT '09. pp. 104–109. Washington, DC, USA (2009)
15. Parra, C., Blanc, X., Duchien, L.: Context awareness for dynamic service-oriented product lines. In: Proc. of SPLC '09. pp. 131–140. Pittsburgh, PA, USA (2009)
16. Rouvoy, R., Barone, P., Ding, Y., Eliassen, F., Hallsteinsen, S., Lorenzo, J., Mamelli, A., Scholz, U.: Music: Middleware support for self-adaptation in ubiquitous and service-oriented environments. In: Software Engineering for Self-Adaptive Systems, Lecture Notes in Computer Science, vol. 5525, pp. 164–182 (2009)
17. Zhang, J., Cheng, B.H.C.: Model-based development of dynamically adaptive software. In: Proc. of ICSE '06. pp. 371–380. ACM, New York, NY, USA (2006)

Parametrización de las transformaciones horizontales en el modelo de herradura

Jesús Sánchez Cuadrado[1], Orlando Avila García[2],
Javier Luis Cánovas Izquierdo[3], Adolfo Sánchez-Barbudo Herrera[2]

[1] Universidad Autónoma de Madrid (jesus.sanchez.cuadrado@uam.es)
[2] Open Canarias, S.L. (orlando@opencanarias.com, adolfosbh@opencanarias.com)
[3] AtlandMod, École des Mines de Nantes – INRIA – LINA, Francia
(javier.canovas@inria.fr)

Resumen En los procesos de modernización o reingeniería de software se aplica generalmente el denominado *modelo de herradura*. En este modelo hay transformaciones verticales entre artefactos software de diferente nivel de abstracción y transformaciones horizontales en el mismo nivel de abstracción. A pesar de ser un modelo conocido y usado en numerosos trabajos todavía no se ha explorado completamente cómo automatizarlo. En este artículo se discuten los problemas existentes para su automatización, y se motiva la problemática con un caso real de modernización. Se describe una aproximación basada en la parametrización de transformaciones horizontales con información descubierta en las transformaciones verticales y los cambios realizados en los modelos de niveles superiores.

1. Introducción

En los procesos de modernización o reingeniería de software se aplica generalmente el denominado *modelo de herradura* [1], un marco de trabajo para integrar los diferentes niveles de abstracción y las actividades que caracterizan este tipo de procesos. La Modernización Dirigida por la Arquitectura (*Architecture-Driven Modernization*, ADM) [2] es una iniciativa de la OMG (*Object Management Group*) que, basándose en el paradigma del Desarrollo de Software Dirigido por Modelos (DSDM), busca proponer y estandarizar técnicas, métodos y herramientas para la modernización en el marco del *modelo de herradura*.

El objetivo principal del modelo de herradura es obtener una representación abstracta del sistema origen que facilite su comprensión y transformación para obtener el sistema destino. Al aplicar este modelo se pueden identificar transformaciones verticales entre artefactos software de diferente nivel de abstracción y transformaciones horizontales en el mismo nivel de abstracción. El modelo no prescribe que las transformaciones sean automatizadas, aunque en ADM se promueve su automatización utilizando técnicas del DSDM.

A pesar de que el modelo de herradura cuenta con un importante bagaje dentro de la comunidad científica y el respaldo de numerosos proveedores de herramientas de modernización, no existen técnicas contrastadas para su automatización ni herramientas que den un soporte integral al modelo. El único

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 551-556, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

punto del modelo en herradura que parece formalmente resuelto es la "extracción" de los llamados "modelos de aplicación" a partir del sistema origen, donde la disciplina de la ingeniería inversa ha trabajado durante décadas. Sin embargo, el resto de etapas permanecen escasamente resueltas.

Algunas aproximaciones tratan de aplicar las transformaciones horizontales, encargadas de la reingeniería, únicamente a alto nivel de abstracción y, a continuación, utilizan transformaciones verticales para generar los artefactos software (ingeniería directa). Como se argumentará en la Sección 2, esta aproximación no es realista porque la información requerida para regenerar el sistema no está disponible en los modelos de más alto nivel de los que se parte.

Este problema se pone de manifiesto en el caso de estudio presentado en la Sección 3, el cual aplica una refactorización del código donde la información de bajo nivel del sistema origen debe prevalecer en el sistema destino, por lo que una aproximación como la aplicada normalmente no es posible. En su lugar, se planteó realizar una transformacón horizontal a bajo nivel de abstracción, pero parametrizada con información de los modelos de más alto nivel.

Nuestro objetivo a largo plazo es estudiar cómo automatizar la aplicación del modelo de herradura. En particular, en este trabajo nos centramos en cómo parametrizar las transformaciones horizontales, ya que pensamos que la clave de la automatización del modelo está en este tipo de transformaciones, puesto que permiten manejar la información descubierta en el proceso de ingeniería inversa a diferentes niveles de abstracción. La Sección 4 presenta dos aproximaciones diferentes, que son aplicables a escenarios de refactorización y migración.

2. El modelo de herradura

En un proceso de reingeniería el objetivo es transformar un sistema software para que cumpla los nuevos requisitos. El modelo de herradura [1] ofrece un marco de trabajo para la aplicación de procesos de reingeniería, facilitando la integración de los diferentes niveles de abstracción involucrados. En este modelo se definen dos tipos de transformaciones: (1) las verticales, que se aplican entre artefactos software de diferente nivel de abstracción, y (2) las horizontales, que se aplican en el mismo nivel de abstracción. La Figura 1a muestra una visión general del modelo, tal y como es propuesto originalmente en [1].

El modelo divide el proceso de reingeniería en tres fases: (1) fase de ingeniería inversa, donde se aplican transformaciones verticales que recuperan las decisiones de diseño tomadas en cada nivel de abstracción; (2) fase de reestructuración, donde se aplican transformaciones horizontales para adaptar o migrar los artefactos existentes al nuevo sistema; y (3) fase de ingeniería directa, donde normalmente se aplica un desarrollo basado en la arquitectura. Es importante observar que la definición del nuevo sistema implica refinar los artefactos de más alto nivel (para añadir nuevas características), pero también integrar los artefactos existentes de bajo nivel a través de transformaciones horizontales, que de alguna forma deben ser informadas por los modelos de nivel superior.

Sin embargo, el modelo normalmente es mal interpretado y simplificado, como se muestra en la Figura 1b. En estos casos se considera que las transformaciones horizontales de la fase de reestructuración se aplican solamente a los

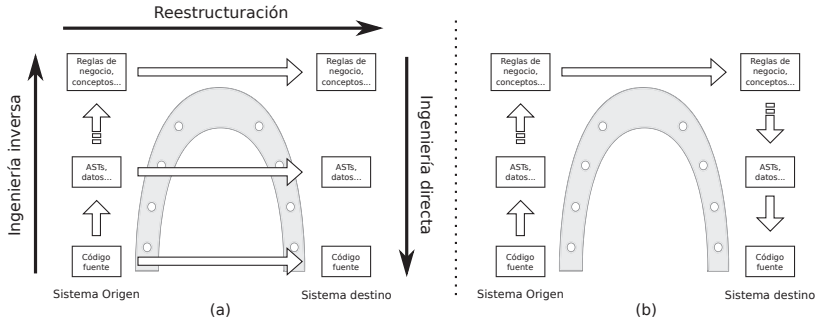


Figura 1. Modelo de herradura (a) original y (b) sin transformaciones horizontales.

artefactos de más alto nivel de abstracción, y que los artefactos resultantes son utilizados para generar el sistema mediante ingeniería directa. Este es el proceso aplicado en trabajos como [3,4] y recomendado por ADM en [5]. Sin embargo, esto no es factible en muchos casos de reingeniería (la sección 3 presenta un ejemplo concreto). El principal problema es la pérdida de información que se produce en las transformaciones verticales (las decisiones de diseño recuperadas en un nivel se pierden al abstraer al nivel superior), que imposibilita "bajar por el lado derecho de la herradura" mediante transformaciones automáticas. Denominaremos a esta versión del modelo "modelo de herradura directo".

Nuestro objetivo es idear un mecanismo basado en técnicas del DSDM para la automatización de procesos de reingeniería donde se consideren las transformaciones horizontales a diferentes niveles de abstracción, respetando el modelo de herradura original. De esta forma, la información recogida en los modelos de más alto nivel, obtenidos tanto en la ingeniería inversa como en la restructuración, guiarán las transformaciones horizontales de los niveles inferiores. El principal reto, por tanto, es cómo parametrizar estas transformaciones horizontales.

3. Caso de estudio

En 2010, el Grupo Banca Cívica decide modernizar su "core bancario" o plataforma tecnológica básica de una entidad financiera. Este escenario requería convertir los programas en "neutros", esto es, independientes de la entidad financiera. Para ello la refactorización debía eliminar cualquier tipo de literal (*hard-coded*) que hiciera referencia al nombre o identificador para convertirlo en una variable que los programas recibieran en tiempo de ejecución.

Las actuaciones a-priori eran sencillas: quitar ciertos literales del código fuente para sustituirlos por variables. Sin embargo, existía un problema importante: el nombre y expresión de la variable por la que que sustituir cada literal dependía del contexto y tipo de programa donde aparecía. Además, existían al menos seis tipos de programas y más de diez contextos de aparición. El proceso de identificación y sustitución dependiente del contexto debía realizarse para las cerca de 100 aplicaciones bancarias, constituidas por más de 14 mil programas COBOL, que sumaban un total de 26 millones de líneas de código fuente.

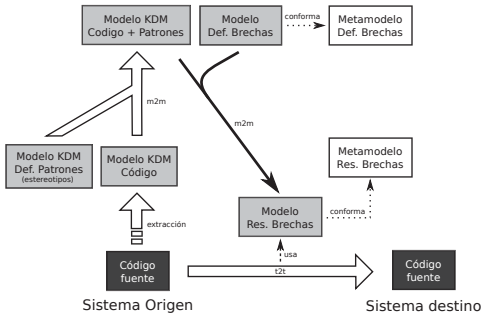


Figura 2. Proceso de transformación llevado a cabo en el proyecto con Banca Cívica

3.1. Aplicación del modelo de herradura directo

El modelo de herradura propone una abstracción progresiva de la descripción de las aplicaciones, de manera que en cada nivel nos podamos centrar en ciertas características (decisiones de diseño) mientras eliminamos otras. Para nuestro problema se percibía necesario abstraer el nivel textual para alcanzar un nivel superior que nos permitiera realizar el *pattern-matching* de literales y sus contextos de aparición, esto es, era necesario un proceso de ingeniería inversa.

Una vez identificadas las apariciones que debían ser sustituidas, había que modificar los programas originales. En este punto, el modelo de herradura directo, tal y como se muestra en la Figura 1b (ingeniería inversa, reestructuración e ingeniería directa) no era factible. Por un lado, debía mantenerse intacto el texto de todo el código no directamente involucrado en la transformación de dichos literales, como los comentarios o las columnas. Por otra parte, incluso sin estas restricciones, regenerar el sistema original siguiendo el modelo directo implicaba trasladar todas las decisiones de diseño e implementación recuperadas del sistema original a través de los diferentes niveles de abstracción. Esta aproximación no es ni eficaz ni eficiente, y por tanto fue descartada.

3.2. Modelo de herradura con parametrización

La solución que se planteó fue hacer una transformación a bajo nivel de abstracción, de texto a texto, de sólo aquellos elementos relevantes. Para conseguirlo, hubo que informar (parametrizar) la transformación con datos inferidos a un mayor nivel de abstracción. Estos datos fueron básicamente dónde y qué tipo de actuación (cambio) realizar en el texto del códido fuente. Esta variación del modelo clásico de herradura se plasma en la Figura 2.

El primer paso fue extraer modelos de código de los ficheros fuente, conformes al metamodelo *KDM* (Knowledge Discovery Metamodel). A partir de este modelo se realizó un nuevo análisis que permitía identificar aquellos literales de interés que se manifestaban en alguno de los contextos definidos, localizando en dichos modelos, las apariciones de los patrones (literal-contexto) planteados.

A continuación se definió el modelo de definición de brechas (*MDB*), que permitía establecer el conjunto de acciones necesarias a acometer sobre el código

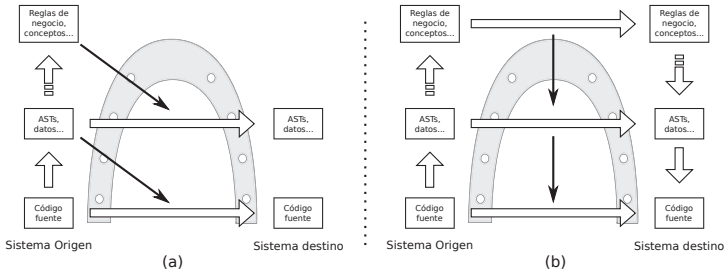


Figura 3. Modelo de herradura con parametrización (a) explícita e (b) implícita.

fuente (nivel de abstracción inferior) para eliminar la aparición de un patrón dado en el modelo de literales identificados (nivel de abstracción superior), esto es, sustituir el literal por la variable correspondiente. De esta forma, por medio de una transformación de modelos, se analizaba el modelo con los literales identificados y en base a lo establecido en el *MDB* se generaba el denominado modelo de resolución de brechas (*MRB*) que es un modelo que especificaba las acciones a realizar por la transformación horizontal de bajo nivel encargada de obtener un fichero fuente con los literales originales sustuidos por variables. El *MRB* se convirtió por tanto en el parámetro para una transformación horizontal a nivel del texto fuente de los programas.

4. Parametrización de las transformaciones horizontales

Dada la problemática explicada en las secciones anteriores, nuestro objetivo es estudiar las diferentes maneras de parametrizar las transformaciones horizontales, permitiendo aumentar el grado de automatización del modelo de herradura. Hasta el momento hemos identificado dos escenarios diferentes.

Escenario 1: Parametrización explícita. Este escenario corresponde al caso de estudio, y la Figura 3a muestra una generalización de este esquema. Como se puede observar, no se producen transformaciones horizontales en los modelos de más alto nivel, sino que éstos sirven para deducir información (de forma automática o asistida por el usuario), que guiará las transformaciones en los niveles inferiores. Para ello, se generan modelos de parámetros (*MRB* en el caso de estudio) utilizando transformaciones modelo-modelo.

Creemos que esta aproximación será útil en escenarios de modernización que no producen cambios en la arquitectura o funcionalidad del sistema. La principal desventaja, es que introduce un nivel adicional de complejidad al tener que definir nuevos metamodelos para los parámetros, así como la creación de nuevas transformaciones que actúan transversalmente. A cambio, es explícito de dónde proviene la información necesaria para las transformaciones horizontales.

Escenario 2: Parametrización implícita. En este escenario sí existen transformaciones horizontales en los modelos de más alto nivel, por ejemplo para realizar cambios en la arquitectura o en la funcionalidad del sistema. Consid-

erese por ejemplo una migración de código Cobol a Java, en la que se pasa de un estilo procedural a uno orientado a objetos.

En este tipo de escenarios los cambios realizados por transformaciones en un nivel de abstracción superior deben tenerse en cuenta en las transformaciones de niveles inferiores para que éstas sean coherentes. Llamamos a este tipo de parametrización *implícita* puesto que debe derivarse del resultado una transformación de modelos (las trazas), como se muestra en la Figura 3b.

En este trabajo no se ha incluido ningún ejemplo de este tipo de escenario, sin embargo los casos de estudio presentados en [6] y [7] sugieren que tiene sentido su aplicación. En ellos los modelos de nivel superior contienen referencias hacia los elementos de nivel inferior. El problema de esta aproximación es que la gestión de las referencias debe realizarse manualmente tanto en las transformaciones verticales como en las horizontales. Además, los metamodelos deben modificarse para tener en cuenta estas referencias.

5. Conclusiones

En este trabajo hemos presentado algunos problemas que aparecen al automatizar el modelo de herradura usando transformaciones verticales y únicamente una transformación horizontal aplicada en el nivel de abstracción superior. Nuestras experiencias en casos de estudio de reingeniería sugieren que el uso generalizado de transformaciones horizontales, a diferentes niveles de abstracción, permiten evitar estos problemas. Sin embargo, estos estudios preliminares también han hecho patente la necesidad de parametrizar dichas transformaciones con información de los modelos de más alto nivel de abstracción.

En este trabajo hemos motivado la problemática, introducido el concepto de parametrización de transformaciones horizontales y presentado una tipología preliminar: explícita e implícita. Para establecer esta tipología nos hemos apoyado en casos de estudio presentados tanto en este trabajo como en trabajos relacionados. Nuestro objetivo a medio plazo es formalizar y definir teoricamente este tipo de parametrización, y proponer métodos y técnicas para incluir transformaciones horizontales en arquitecturas generativas aplicadas a los distintos escenarios de reingeniería de software.

Referencias

1. Kazman, R., Woods, S.S., Carrière, S.J.: Requirements for integrating software architecture and reengineering models: Corum ii. In: WCRE. (1998) 154–163
2. Ulrich, W.M., Newcomb, P.: Information Systems Transformation: Architecture-Driven Modernization Case Studies. Morgan Kaufmann (2011)
3. Kshusidman, V.: Soa enabled workflow modernization. BP Trends (2006)
4. Yu, Y., Wang, Y., Mylopoulos, J., Liaskos, S., Lapouchnian, A., do Prado Leite, J.C.S.: Reverse engineering goal models from legacy code. In: RE. (2005) 363–372
5. Kshusidman, V.: Adm transformation. ADM Task Force. White Paper (2008)
6. Sánchez Ramón, O., Sánchez Cuadrado, J., García Molina, J.: Model-driven reverse engineering of legacy graphical user interfaces. In: ASE '10. (2010) 147–150
7. Sánchez Ramón, O., Sánchez Cuadrado, J., García Molina, J.: Reverse engineering of event handlers of rad-based applications. WCRE (2011) 293–302

Transformación de modelos con Eclectic

Jesús Sánchez Cuadrado[1]

Universidad Autónoma de Madrid (Spain)
jesus.sanchez.cuadrado@um.es

Abstract. Las transformaciones de modelos son un elemento clave del Desarrollo de Software Dirigido por Modelos. En los últimos años se han propuesto varios lenguajes de transformación de diferente naturaleza, siendo cada uno de ellos adecuado para un determinado tipo de tarea de transformación. Sin embargo, una transformación compleja normalmente implica abordar una serie de sub-problemas que corresponden a diferentes estilos de transformación, y por tanto no toda la transformación puede desarrollarse de forma natural en el lenguaje elegido.
En esta demostración se presentará el entorno de transformación de modelos *Eclectic*, que trata de abordar el desarrollo de transformaciones de modelos ofreciendo una familia de lenguajes de transformación. Cada lenguaje tiene como objetivo abordar un determinado tipo de transformaciones, y está específicamente diseñado para ello. La demostración se ilustrará con un ejemplo de aplicación que utiliza diferentes lenguajes, se mostrará el entorno de desarrollo y se comentarán características de la aproximación tales como interoperabilidad entre lenguajes e integración con programas Java.

1 Motivación

Las transformaciones de modelos son un elemento clave del Desarrollo de Software Dirigido por Modelos (DSDM), puesto que permiten automatizar la manipulación de los modelos. En los últimos años se han propuesto varios lenguajes de transformación de diferente naturaleza, tales como ATL [5], RubyTL [3], QVT [8] o Kermeta [6]. Las taxonomías de lenguajes de transformación propuestas por Czarnecki [4] y Mens [7] evidencian que cada lenguaje proporciona una serie de características que lo hacen más adecuado para abordar cierto tipo de tarea de transformación. Hasta el momento el diseño de lenguajes de transformación ha seguido básicamente dos aproximaciones: a) mantener el lenguaje simple, declarativo y orientado a cierto tipo de transformaciones, b) complicar el lenguaje añadiendo características para que tenga un ámbito de aplicación más amplio. El problema de la primera aproximación es que la aplicabilidad del lenguaje es limitada, y normalmente solo permite que el lenguaje se use para abordar transformaciones simples, mientras que la segunda aproximación complica el diseño original con lo que las transformaciones tienen tendencia a ser ilegibles a medida que son más complicadas, debido principalmente a que las características declarativas del lenguaje no son suficientes.

Para abordar esta problemática se propone un diseño alternativo, basado en una familia de lenguajes de transformación [2].

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 557-560, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

2 Eclectic: una familia de lenguajes de transformación

En sección se describe brevemente las características de la familia de lenguajes de transformación que se ha definido, llamada Eclectic, así como su estado actual y el trabajo futuro.

La idea principal de esta aproximación es que una transformación compleja podría dividirse en varias tareas de transformación de tamaño más pequeño, que podrían resolverse utilizando lenguajes específicamente diseñado para ellas. Se pretende así mejorar la declaratividad y la "intencionalidad" de las definiciones de transformación, puesto que los lenguajes que se usarían no son de propósito general, sino que tienen las características (y solo esas) que hacen falta resolver ese tipo de tarea.

Para alcanzar ese objetivo es imprescindible que los diferentes lenguajes que componen Electic sean interoperables, así como disponer de mecanismos de composición que permitan especificar cómo los resultados obtenidos por cada una de las transformaciones contribuyen al resultado final. En estos momentos se han investigado los siguientes elementos de la aproximación:

- *Interoperabilidad* entre lenguajes de transformación heterogéneos. Este objetivo se ha conseguido definiendo un lenguaje intermedio, IDC (Intermediate Dependency Code), al cual compilan el resto de lenguajes. Algo más de información acerca de IDC puede encontrarse en [9].
- *Mecanismos de composición* en transformación de modelos. El lenguaje IDC soporta los siguientes: alimentar reglas o patrones con valores obtenidos mediante cierto recorrido de un modelo, resolución de referencias de elemento origen a destino, decorar metaclases con métodos virtuales y configuración de la ejecución de transformaciones.
- *Tipos de transformaciones*. ¿Qué tipos de transformaciones debería soportar la familia de lenguajes que se está construyendo? Hasta ahora se han identificado cinco dominios y se ha implementado un lenguaje para cada uno de ellos:
  - Definición de correspondencias o *mappings*. Tiene como objetivo resolver heterogeneidades entre partes de meta-modelos semánticamente equivalentes.
  - Transformaciones dirigidas por la estructura del modelo destino. Suelen darse cuando se compila un modelo de más alto nivel de abstracción a otro de más bajo nivel, y requiere inicializar varios objetos a partir de cierto objeto origen.
  - Cálculo de atributos, al estilo de las definiciones dirigidas por la sintaxis [1], donde el valor de un atributo para un elemento depende de los valores de los atributos de otros elementos accesibles a partir del primero.
  - Búsqueda de patrones complejos, cuyo resultado debe utilizarse para alimentar las reglas de algunos de los lenguajes de la familia.
  - Composición de transformaciones, para crear una transformación que aborde el problema global.

– *Integración* con lenguajes de programación de propósito general. Para abordar este objetivo se ha creado un compilador que genera *bytecode* para la máquina virtual de Java (JVM), de manera que las transformaciones escritas con Eclectic puede integrarse de manera natural con programas Java. En este aspecto es importante considerar también la integración a nivel de entorno de desarrollo, que se ha realizado para Eclipse.

La versión actual de Eclectic es todavía una prueba de concepto pero ya se ha conseguido crear una implementación que demuestra que la aproximación es viable. Está disponible en `http://sanchezcuadrado.es/projects/eclectic` para su descarga. La Figura 1 es una captura de pantalla del entorno que se está construyendo para Eclipse, en el que se están editando dos transformaciones, que tratan con modelos UML, OCL y Java: una con el lenguaje de cálculo de atributos y otra con el lenguaje de *mappings*.

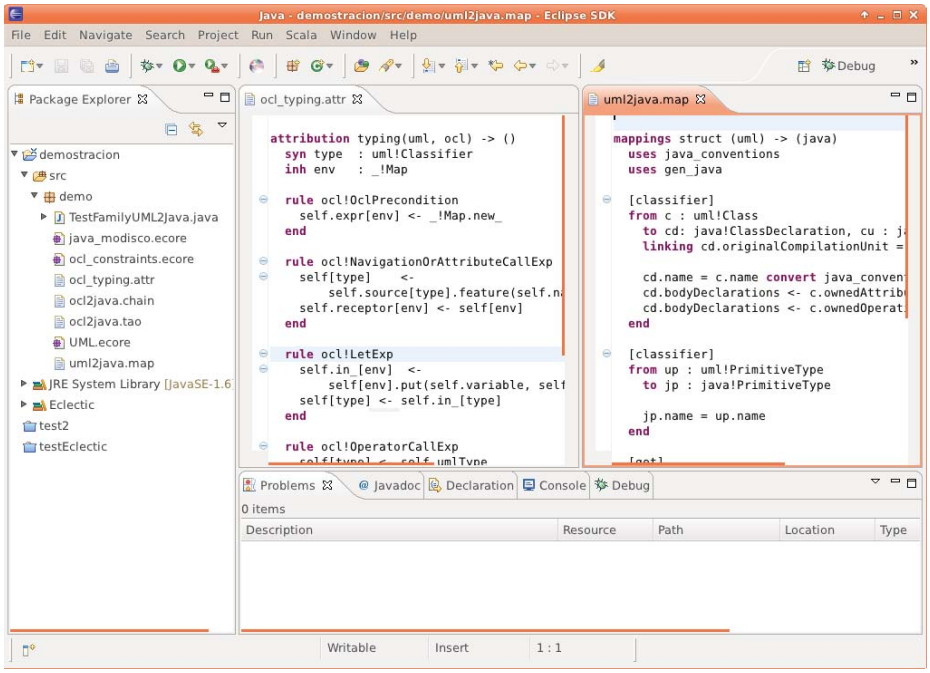


Fig. 1: Entorno de desarrollo en Eclipse

Como trabajo futuro se plantean las siguientes cuestiones:

– Seguir investigando y refinando los lenguajes que componen Eclectic y sus características, así como escribir guías que orienten al desarrollador a la hora de elegir el lenguaje más apropiado y aplicarlo.
– Implementar casos de estudio de relativa complejidad que permitan determinar si esta aproximación presenta una mejora real con respecto a utilizar un único lenguaje de transformación.

– Estudiar la integración de lenguajes de transformación específicos del dominio, posiblemente en forma de librerías.
– Aprovechar las posibilidades de la JVM para mejorar el rendimiento.

3 Demostración

Con la presente demostración se pretende dar a conocer Eclectic y recabar la opinión de la comunidad sobre esta aproximación así como posibles mejoras o ideas. Para ello se prepará un póster que resuma las características de Eclectic y motive su utilización, así como una serie de ejemplos de aplicación para ser mostrados a los interesados en tener más detalles.

Agradecimientos. Este trabajo ha sido financiado por el Ministerio de Educación y Ciencia (TIN2011-24139) y la Comunidad de Madrid (S2009/TIC-1650).

References

1. A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers: principles, techniques, and tools*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1986.
2. J. S. Cuadrado. Towards a family of model transformations languages. In *Proc. of the 5th International Conference on Model Transformati on (ICMT 2012, to appear)*, pages 260–275, LNCS 7307, 2012. Springer.
3. J. S. Cuadrado, J. G. Molina, and M. Menarguez. RubyTL: A Practical, Extensible Transformation Language. In *2nd European Conference on Model Driven Architecture*, volume 4066, pages 158–172. Lecture Notes in Computer Science, June 2006.
4. K. Czarnecki and S. Helsen. Feature-based survey of model transformation approaches. *IBM Syst. J.*, 45:621–645, July 2006.
5. F. Jouault, F. Allilaire, J. Bézivin, and I. Kurtev. ATL: A model transformation tool. *Science of Computer Programming*, 72(1-2):31 – 39, 2008. See also `http://www.emn.fr/z-info/atlanmod/index.php/Main_Page`. Last accessed: Nov. 2010.
6. Kermeta. `http://www.kermeta.org/`.
7. T. Mens and P. Van Gorp. A taxonomy of model transformation. *Electron. Notes Theor. Comput. Sci.*, 152:125–142, March 2006.
8. OMG. Final adopted specification for MOF 2.0 Query/View/Transformation, 2005. www.omg.org/docs/ptc/05-11-01.pdf.
9. J. Sanchez Cuadrado. Compiling ATL with Continuations. In *Proc. of 3rd International Workshop on Model Transformation with ATL (MtATL-2011)*, pages 10–19. CEUR-WS, 2011.

Prueba de Transformaciones de Modelos con TractsTool

Manuel Wimmer, Loli Burgueño, and Antonio Vallecillo

GISUM/Grupo de investigación Atenea. Universidad de Málaga
{mw,loli,av}@lcc.uma.es

Resumen Las tranformaciones de modelos son un elemento esencial en la Ingeniería Dirigida por Modelos (Model-driven Engineering, MDE) y por ello una tarea que está cobrando relevancia es probar su corrección. Los *Tracts* ofrecen un enfoque modular y extensible para la especificación y verificación de transformaciones de modelos. Este trabajo presenta TractsTool, una herramienta desarrollada en Eclipse que implementa los mecanismos que proporcionan los *Tracts*.

Palabras clave: MDE, Transformaciones de Modelos, Tracts

1. Introducción

La Ingeniería Dirigida por Modelos es una metodología que trabaja con modelos de dominio. Una tarea importante en este ámbito es la transformación de modelos, para la cual existen lenguajes que a partir de uno o varios modelos origen permiten obtener uno o varios modelos destino. Cada modelo implicado en una transformación es conforme a un metamodelo, que también debe ser conocido.

En sus inicios las transformaciones de modelos eran simples pero conforme la tecnología y el software ha ido evolucionando las tranformaciones cada vez son más complejas y por tanto aumenta la dificultad de probar su corrección.

Los *Tracts* [1] son un enfoque para la especificación y prueba de transformaciones de modelos. Concretamente, son un conjunto de restricciones impuestas sobre los modelos de entrada, de salida y sobre la relación que deben mantener los modelos de entrada con los de salida. El objetivo es que una vez ejecutada la transformación se comprueben las restricciones y en caso de que se satisfagan se podrá tener un mayor grado de certeza de que la transformación es correcta.

Como se puede intuir, puede ser costoso crear manualmente un número considerable de modelos de entrada para probar todos los casos que se puedan dar. Por ello, se hace uso de un lenguaje de creación de modelos, ASSL (A Snapshot Sequence Language) [2], que genera el conjunto de entrada. Concretamente, con ASSL se especifican las propiedades que los modelos generados (conforme a un metamodelo dado) deben satisfacer.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 561-564, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

2. Objetivos y Enfoque

El objetivo de este trabajo es obtener una herramienta genérica que dada cualquier transformación sea capaz validarla de forma empírica, es decir, basándose en la experiencia de ejecutarla sobre un número considerable de diferentes modelos origen, usando Tracts.

Los elementos que se le deben proporcionar a la herramienta son: (1) la transformación, (2) los metamodelos de entrada y salida, (3) el programa ASSL con el cual se obtiene la colección de modelos escritos en el lenguaje del metamodelo de entrada y (4) los Tracts.

La transformación a probar debe ser escrita en el lenguage ATL [3] y los metamodelos que la herramienta TractsTool soporta deben ser conformes al meta-metamodelo de Ecore.

Las restricciones (Tracts) deben ser una serie de predicados que aclaren si tras la transformación los modelos son válidos o no. Ya se comentó en la sección 1 que pueden ser definidas sobre los modelos de entrada, sobre los modelos de salida y sobre la relación entre ambos. Concretamente, el lenguaje de definición de condiciones es OCL [4].

Respecto a la obtención del juego de modelos de entrada, en la versión actual de la herramienta es necesario proporcionar el código ASSL. Uno de los objetivos futuros es generar automáticamente un programa ASSL a partir del metamodelo de entrada que proporcione un conjunto de modelos por defecto.

3. Implementación

La validación de las restricciones OCL y la generación de los modelos origen hacen uso de la herramienta USE (UML based Specification Environment) [6] [7] por lo que es necesario adaptar las entradas de TractsTool para que puedan ser usadas en USE. Dicha adaptación conlleva la realización de determinadas transformaciones sobre los metamodelos y modelos para que sean comprensibles por USE.

El esquema de la Figura 1 muestra los diferentes pasos que la herramienta ejecuta hasta obtener la solución. En primer lugar, en el paso (1) se transforman los metamodelos de entrada a los metamodelos conforme al meta-metamodelo de USE. A continuación, con la herramienta USE, en el paso (2) se crea el conjunto de modelos de entrada que en el paso (3) se transforman al formato Ecore. En (4) se aplica la transformación ATL que se quiere probar y se obtienen los modelos destino, los cuales en (5) son transformados al formato de USE. Por último, en el paso (6) son validados los Tracts en USE, obteniendo como salida si son verdaderos o falsos y en caso de que sean falsos las instancias que lo han provocado.

En la Figura 2 se puede ver la GUI de TractsTool tras una ejecución. En este caso concreto la transformación es Families2Persons. Un ejemplo detallado y explicado del código ATL, el código ASSL, los Tracts, etc. se puede encontrar en el informe técnico accesible en http://atenea.lcc.uma.es/Descargas/F2P.pdf

En la parte derecha de la interfaz gráfica se puede apreciar que aparecen todas las invariantes definidas en los Tracts, junto con la información de si se satisfacieron o no y en caso negativo el conjunto de instancias que lo hicieron fracasar.

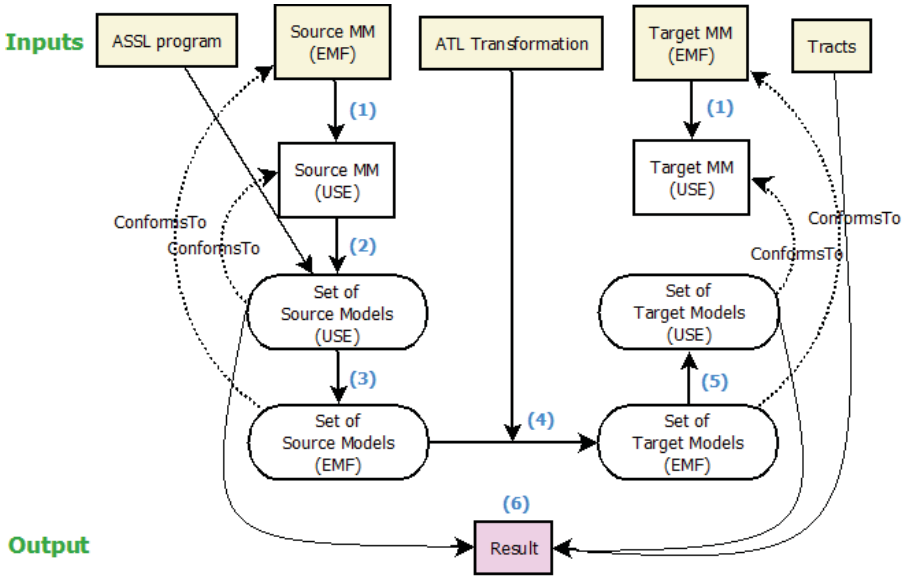


Figura 1. Esquema de implementación

La herramienta está creada como un plugin de Eclipse. Para evitar incompatibilidades, se puede descargar el Eclipse completo con el plugin integrado desde la web de la herramienta (http://atenea.lcc.uma.es/index.php/Main_Page/Resources/Tracts). Además en dicha web aparecen los requisitos para la instalación, un video donde se ve la herramienta en ejecución y ejemplos para descargar.

4. Conclusiones y Trabajo Futuro

Se ha presentado la herramienta TractsTool que permite comprobar la corrección de las transformaciones ATL a través de la validación de restricciones OCL sobre los modelos de entrada y/o salida.

Actualmente se ha descrito esta primera versión de la herramienta pero se prevee continuar añadiendole funcionalidad. Uno de los objetivos es generar automáticamente código ASSL que partir del metamodelo de entrada dé lugar a modelos por defecto, liberando al usuario de la tarea de programar en este lenguaje. Además se pretende modificar la salida de TractsTool que actualmente es texto y que pasara a devolver modelos de diagnóstico. Otro trabajo futuro es hacer que la herramienta sea capaz de probar transformaciones de modelos escritas en otros lenguages como QVT [5].

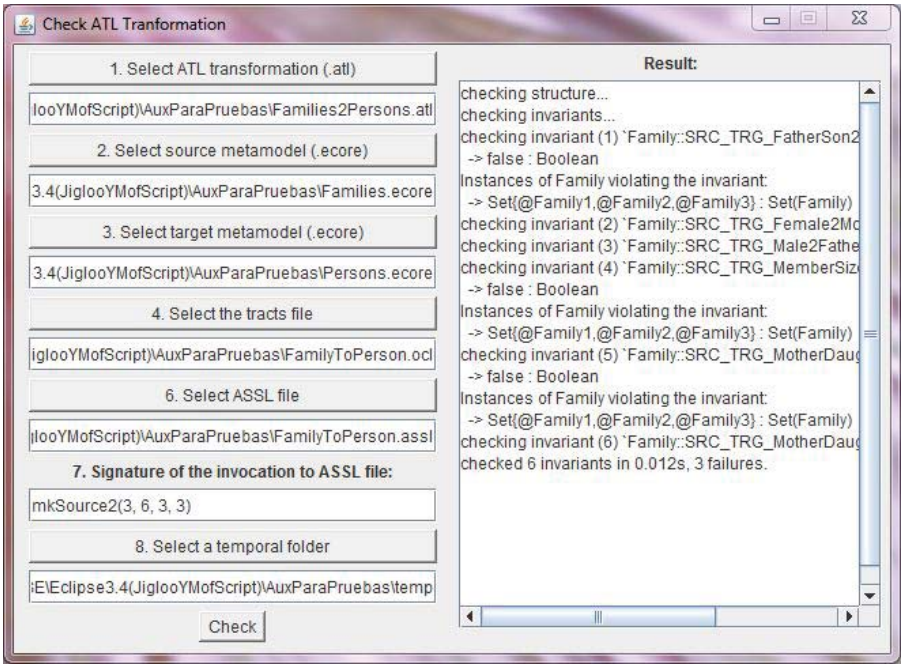


Figura 2. Interfaz gráfica de TractsTool

Agradecimientos: A los revisores por su dedicación para proponer mejoras a este trabajo y al Ministerio de Ciencia e Innovación por haber financiado los proyectos TIN2011-23795 y TIN2008-03107 en cuyo marco se ha desarrollado el trabajo.

Referencias

1. Gogolla, M., Vallecillo, A.: Tractable model transformation testing. In: Proc. of the Seventh European Conference on Models Foundations and Applications (ECMFA 2011). Volume 6698 of LNCS., Birmingham, UK, Springer (2011) 221–236
2. Gogolla, M., Bohling, J., Richters, M.: Validating UML and OCL Models in USE by Automatic Snapshot Generation. Software and Systems Modeling **4**(4) (2005) 386–398
3. Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I.: ATL: A model transformation tool. Sci. Comput. Program. **72**(1-2) (2008) 31–39
4. Cabot, J., Gogolla, M.: Object Constraint Language (OCL): the Definitive Guide. In: Formal Methods for Model-Driven Engineering. LNCS, Bertinoro, Italy, Springer (2012)
5. OMG: MOF2.0 query/view/transformation (QVT) adopted specification. (OMG document ptc/05-11-01, 2005. available from www.omg.org.)
6. Richters, M., Gogolla, M.: On formalizing the uml object constraint language ocl. In: ER. (1998) 449–464
7. Richters, M., Gogolla, M.: Ocl: Syntax, semantics, and tools. In: Object Modeling with the OCL. (2002) 42–68

Desarrollo dirigido por modelos de visualización de datos para la Web

Rober Morales-Chaparro, Juan Carlos Preciado, Fernando Sánchez-Figueroa

Quercus Software Engineering Group, Universidad de Extremadura*
{robermorales,jcpreciado,fernando}@unex.es

Resumen La visualización de datos juega un papel clave al presentar informes empresariales. Una adecuada visualización sirve de soporte para fundamentar mejor la toma de decisiones. Sin embargo, una visualización desacertada puede inducir a tomar una decisión incorrecta. Habitualmente, la elección de diferentes representaciones visuales para un mismo conjunto de datos se prefija por los desarrolladores de la misma. En este contexto, el usuario final no tiene ocasión de ajustar o cambiar las visualizaciones en tiempo de ejecución y no todos los usuarios tienen por qué tener los mismos intereses cuando visualizan los mismos datos. En este trabajo presentamos un lenguaje específico de dominio que permite definir semi-automáticamente patrones de visualización reutilizables de manera que el usuario final pueda elegir el que más se adapte a sus intereses.

Palabras clave: Visualización de datos, Desarrollo de software dirigido por modelos, Lenguajes específicos del dominio, Ingeniería Web

1. Introducción

Las empresas disponen cada vez más de un gran volumen de datos sobre su actividad, habitualmente con formatos y fuentes heterogéneos. Todo ello en un contexto donde además la cantidad de datos disponibles en Internet provoca una sobrecarga de información que hace aún más compleja la toma de decisiones, piezas clave en los emergentes sistemas de Business Intelligence. Por tanto, los usuarios necesitan más y mejores formas de acceder a los mismos. En estos procesos de información, los datos necesitan ser explorados, pero también analizados, interpretados o incluso comunicados a otras personas. En este escenario, las modernas técnicas de visualización de datos se conforman como una herramienta útil que permite simplificar visualmente estos procesos de consulta[1].

Sin embargo, el diseño y desarrollo de visualizaciones de datos no es una tarea fácil, teniendo que hacer frente a diferentes requisitos. Por un lado, a) por cuestiones tecnológicas: la evolución de las capacidades de reproducción visual (como la aparición de HTML5), los diferentes dispositivos de visualización (teléfonos, tablets,...), las fuentes de datos heterogéneas (lenguajes, APIs,...), etc.,

* Este trabajo ha sido financiado por el proyecto MIGRARIA - TIN2011-27340 del Ministerio de Ciencia e Innovación y por el Gobierno de Extremadura.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 565-578, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

Estos son problemas recurrentes en Ingeniería del Software. y, por otro lado, b) a nivel del usuario: intentar que la visualización cubra las expectativas de los usuarios en función del contexto (perfil de usuario, intereses, hábitos de los sujetos similares, tendencias del mercado, tendencias sociales, etc.). Que el sistema disponga de las capacidades necesarias para que el usuario pueda interpretar los datos de forma visual y certera es una pieza clave en los sistemas de Business Intelligence. En un escenario ideal, la visualización de un conjunto de datos debe tener en cuenta los dos grupos de requisitos. Debería poder desarrollarse para diferentes plataformas y dispositivos y hacer uso de las mejores tecnologías disponibles en cada momento. Además, la misma información debe poder ser mostrada de diferentes formas atendiendo al cambiante contexto del usuario.

[2] ofrece una solución a ambos problemas sobre la base de un proceso dirigido por los datos y por el usuario. Así, las cuestiones tecnológicas son tratadas orientando el desarrollo mediante una perspectiva dirigida por modelos (MDE), mientras que los aspectos sociales y de interacción se encaran permitiendo al usuario crear, utilizar y personalizar las presentaciones para obtener la información en un formato visual de la manera que el usuario considera más apropiada en cada momento. En este trabajo se presentan más detalles del proceso descrito en [2], centrándonos aquí en el modelado de visualizaciones. El modelado del proceso se realiza a través de un lenguaje específico de dominio (DSL) que permite al usuario experto involucrarse junto con el modelador en los desarrollos y/o disponer de capacidades para modificarlos más tarde, en tiempo de ejecución. Lejos de dar todos los detalles, el trabajo muestra los principales pilares e ilustra su uso a través de un ejemplo. Este DSL aporta dos beneficios principales. El primero, la posibilidad de que el diseñador reutilice diferentes patrones de visualización entre diferentes aplicaciones, independientemente de que se quiera generar código para diferentes tecnologías, plataformas o dispositivos. El segundo, la oportunidad de que el usuario disponga de varias visualizaciones distintas para el mismo conjunto de datos. El procedimiento detallado se circunscribe al ámbito de la visualización explorativa y analítica. Otros tipos de visualización como la colaborativa o la narrativa aún no se han tratado dentro de este trabajo.

El resto del artículo se estructura como sigue. La sección 2 muestra un ejemplo motivador. La sección 3 describe un enfoque global, mientras que en la sección 4, 5 y 6 se detallan las fases principales de la propuesta (fuentes de datos, modelado de la visualización y ejecución). Por último, revisaremos trabajos relacionados en la sección 7, y las conclusiones y el trabajo futuro en la sección 8.

2. Ejemplo conductor

Una empresa española con centros localizados en diferentes puntos de la península desea mejorar la calidad en su atención al cliente. El director de calidad quiere comparar las reclamaciones recibidas durante los últimos años en sus tiendas para detectar posibles problemas. La empresa cuenta con todos los datos necesarios sobre este tema, aunque el director todavía no sabe qué decisión va a tomar, quiere visualizarlos de forma exploratoria. El sistema informático que

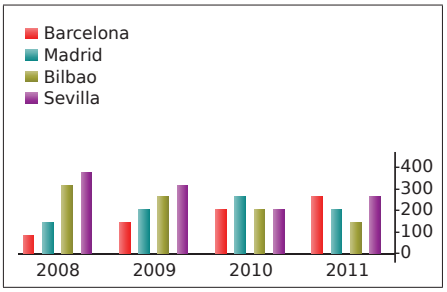


Figura 1. Visualización elegida por el desarrollador.

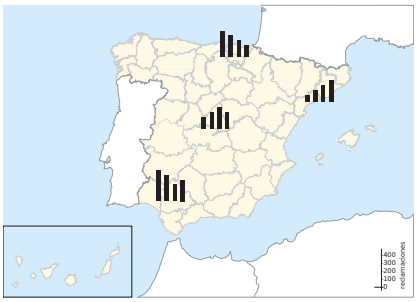


Figura 2. Visualización preferida por el director de calidad.

utilizan muestra estos datos con una visualización, Fig 1, que obedece a un patrón visual elegido por el desarrollador del sistema para este tipo específico de datos.

Este gráfico prioriza la visión general con respecto al detalle, por lo que observando el gráfico se puede interpretar que el número de reclamaciones no ha aumentado con el tiempo y se mantiene más o menos constante entre 150 y 250 por tienda y año. De hecho, el director de calidad aprecia que el número de reclamaciones por año es cada vez más similar entre todas las tiendas y no puede sacar ninguna conclusión útil.

Esta representación no ofrece un seguimiento visual apropiado de la localización de los centros, ya que utiliza la misma variable visual (la posición en x) para dos variables léxicas (la tienda y el año). Desde el punto de vista del director, priorizar la secuencia a partir de este dato puede ser relevante, y le hubiera gustado obtener la visualización que se muestra en la Fig. 2.

El cambio juega un papel relevante en la conclusión. Utilizando la nueva visualización, se comprende rápidamente que el comportamiento no es igual en todas las tiendas. Las hay donde la calidad mejora (las reclamaciones bajan), otras donde es más estable en el tiempo, y sobre todo, una tienda donde las reclamaciones suben año tras año (Barcelona).

Sin cambiar los datos, y tan sólo *dependiendo de la visualización utilizada*, podría haber tomado una decisión, o la contraria. El problema proviene del hecho de determinar a priori el tipo de visualización que se utilizará, sin ninguna posibilidad de ajustar o cambiar la misma. ¿Por qué no representar los datos visualmente de acuerdo a las necesidades del director? y, es más, ¿por qué no dejar que él la cambie de acuerdo con sus expectativas?

A continuación mostramos brevemente el proceso que soporta esta posibilidad y sus diferentes fases, detallando especialmente el modelado de las visualizaciones.

3. Proceso propuesto

La Fig. 3 muestra una visión general de la arquitectura del sistema. El punto de entrada corresponde fundamentalmente a los datos. El razonador genera un

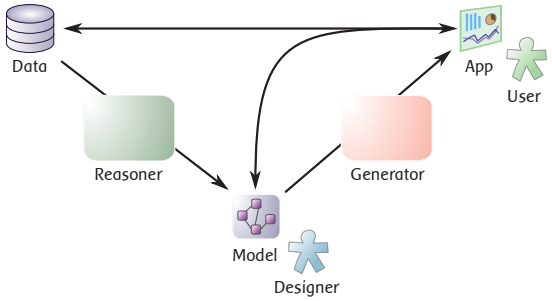


Figura 3. Visión general.

modelo de visualización predeterminada de los mismos, que el modelador puede refinar. Este modelo sirve para generar el código de la aplicación final. El usuario puede cambiar de un patrón a otro buscando el cumplimiento de sus expectativas. El usuario también puede colaborar en la edición o creación de patrones ya que el DSL está orientado a los conceptos de la visualización y por tanto no exige grandes conocimientos técnicos para su uso.

La descripción de los datos es la expresión explícita de la estructura de datos y de su origen, conforme a un esquema formal. Los patrones de visualización son desarrollados por los modeladores con la ayuda de un sistema razonador que busca situaciones similares en el pasado. Para este propósito, se ha definido un lenguaje específico de dominio (DSL). El DSL permite que el generador produzca código a partir de los modelos, el cual se conecta con los datos, ejecuta transformaciones sobre ellos y muestra el resultado final.

A continuación, narramos las diferentes fases del proceso más en detalle.

4. Datos. Especificación de las fuentes de datos y sus descripciones.

Por cada conjunto de datos que se use como fuente en una visualización, se necesitan seleccionar algunos parámetros. El primero es la ubicación de origen de datos, que es una dirección URL. Posteriormente, se incluye una referencia a la representación metalenguaje (XML, JSON, YAML, CSV, XLS, ODS,...) o la codificación (UTF-8, Latin-15,...).

A continuación, debemos seleccionar el nombre y los atributos de cada tabla del conjunto de datos. Cada atributo tiene varios detalles: `nombre` y `tipo` son obligatorios a fin de realizar una conexión exitosa en tiempo de ejecución. Otros meta-datos pueden especificarse también, para mejorar la comprensión del sistema de la fuente (y conducir la selección automática tanto como sea posible). Estos pueden incluir la descripción, `null` (si los valores vacíos están permitidos), el valor neutro para rellenar los huecos, si es clave, o identificador, o etiqueta, o clave externa,...

Un conjunto de datos de muestra se presenta en el Cuadro 1.

Cuadro 1. Datos de ejemplo.

Año: integer	Ciudad: string	Reclamaciones: integer
2008	Barcelona	101
2009	Barcelona	152
2010	Barcelona	199
...	...	...

5. Desarrollo dirigido por modelos de los patrones de visualización.

El objetivo de esta sección es mostrar cómo se desarrollan los modelos basándose en flujos estructurados, que transforman los datos en variables retinales, para construir visualizaciones. Esto se consigue utilizando un lenguaje específico del dominio que es la principal contribución de este trabajo y que detallaremos en esta sección. En primer lugar, presentaremos los fundamentos teóricos del DSL. Después describiremos la biblioteca nativa utilizando el DSL para construir el patrón de la Fig. 1.

5.1. Bases

Tres conceptos clave constituyen el núcleo del DSL.

Los nodos procesan, y los flujos comunican los datos. El lenguaje se basa en que hay nodos que reciben una o varias entradas de otros nodos transformándolas en una o varias salidas realizando algunos cálculos. Las salidas están conectadas a otras entradas de nuevo, utilizando los flujos para encadenar la ejecución de nodos.

Las entradas deben ser del tipo esperado. Existen operaciones de conversión entre los valores de diferentes tipos, pero en esencia las salidas de un nodo sólo pueden utilizarse como entrada de otros si son *del mismo tipo*.
Esta función permite que los módulos que tengan las mismas entradas y salidas sean permutables. Así, los patrones visuales serán directamente intercambiables entre sí cuando reciben las mismas entradas (ya que la salida es siempre la misma, un dibujo).

Los nodos esperan *sólo* por sus datos. El orden de ejecución no está limitado ni restringido. La única condición es que un nodo no se ejecutará antes de que todas sus entradas estén disponibles.
Esto permite que el modelador piense en el modelo como en un árbol de dependencias, no como en una lista de tareas paso a paso. El dominio, principalmente declarativo, basado en transformaciones, sugiere que este es un buen enfoque.

Los detalles gramaticales más relevantes del DSL son:

Modularidad y unidades de traducción. Cada concatenación de elementos puede ser empaquetada como un módulo. Esto hace que los módulos sean reutilizables (y compartibles) tanto entre proyectos como dentro del mismo. Cada módulo está en un archivo separado, y viceversa.

Persistencia. Los módulos se almacenan en archivos basados en texto que se ajusten a una gramática formal. Un nivel sintáctico gráfico servirá de vista alternativa, para ayudar al modelador en varias tareas: comunicación, documentación, aprendizaje,...

5.2. Librería nativa

Una biblioteca nativa es básica para tener un conjunto básico de tipos y operaciones. Esto mejora la compatibilidad futura y proporciona una experiencia de desarrollo mejor. Todos los modelos pueden utilizar los elementos de la librería nativa. En esta sección explicaremos esta librería desarrollando un patrón a modo de ejemplo, el de la Fig. 1. Como todas las visualizaciones, tiene una geometría subyacente, descrita en la Fig. 8. Los tipos nativos se dividen en tres categorías. El modelador puede crear nuevos tipos sobre la base de los ya existentes.

Tipos léxicos. Orientados principalmente a clasificar los datos procedentes de fuentes externas: `Bool, Integer, Float, String`. Entre ellos, los tipos léxicos se pueden enriquecer indicando la densidad del espacio: categórica, ordinal o cuantitativa.

Tipos visuales. Tienen como intención representar las variables retinales [3]. La mayoría de los lenguajes de propósito general no soportan como primitivas esos conceptos. `Posición, tamaño, ángulo, color, forma, textura`.

Tipos estructurales. Pensados para combinar, agrupar y organizar otros tipos de datos: `sequence` (listas de valores del mismo tipo) y `tuple` (diccionarios, un almacenamiento de claves-valor de tipos heterogéneos).

Para los tipos nativos se pueden expresar valores constantes de forma textual, ya que se proporcionan literales. A continuación, se explican las operaciones nativas que también se clasifican para una mejor comprensión de su propósito.

Accesores. Leen y/o escriben los atributos de las tuplas. Visualmente, están representadas por una caja con un par de corchetes.

Constructores. Pensados para crear nuevas instancias de objetos a través de la asignación de valores a los atributos necesarios. La biblioteca nativa proporciona constructores para los tipos básicos, principalmente utilizados para generar formas. Las invocaciones de los constructores son, visualmente, cajas con iconos y entradas etiquetadas.

Con las herramientas explicadas, podemos empezar ya a desarrollar el modelo detrás de la Fig. 1. El punto de partida seleccionado es un patrón que construya una sola barra, cada vez que se invoque. Lo primero a hacer para su desarrollo será descubrir las relaciones geométricas detrás de cada barra. Esto nos ayudará a generar una implementación orientada a los resultados.

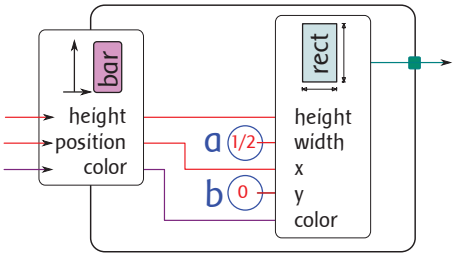


Figura 4. Ejemplo de patrón reutilizable que construye una barra.

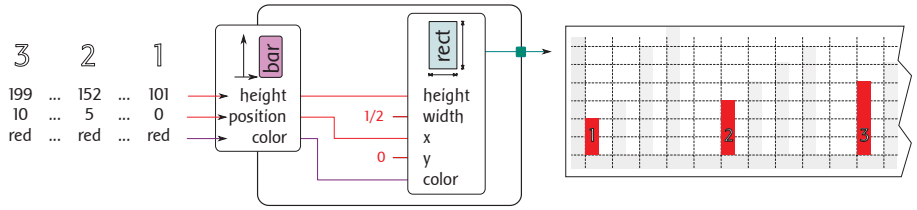


Figura 5. Invocaciones de la Fig. 4.

En este caso, vamos a seguir la descripción de la Fig. 8, que está basada en un sistema de coordenadas de dos dimensiones. Cada barra es básicamente un rectángulo, y hay dos invariantes para todas ellas: `(a)` *width* $= 1/2$ y `(b)` $y = 0$.

Por lo tanto la principal tarea del patrón, que se muestra en la Fig. 4, es incluir un *constructor* de la primitiva rectángulo. Los hechos `a` y `b` se implementan explícitamente con el uso de constantes. Por otra parte, los valores variables (altura, posición-x y color) se toman de las entradas mientras que la salida es la forma construida (el rectángulo).

Ofrecemos en la Fig. 5 una simulación para una ejecución del patrón, repetida tres veces. Los primeros valores de entrada para el patrón (que se muestran a la izquierda) van construyendo algunos de los primeros rectángulos de la primera serie de la Fig. 8 (etiquetados en la Fig. 5 como `1,2` y `3`).

Podemos ver como ejemplo cómo la primera barra de la primera serie está en la posición 0, es de 1 unidad de altura, y es de *color* rojo. Las barras consecutivas se irán construyendo a medida que lleguen los datos a los flujos de entrada. Para seguir construyendo el modelo del gráfico de barras necesitamos conocer más operaciones.

Operaciones básicas de manipulación. Aquí se encuentran las operaciones que manipulan cadenas: *split*, *concat*, *substring*,... Expresiones regulares: *test*, *match*, *replace*,... Operaciones algebraicas: *add*, *subs*, *times*, *quotient*, *modulus*,... Así como con acumuladores y operadores estadísticos: *size*, *average*, *maximum*, *variance*, *count*, *sum*,...

La apariencia visual de estos manipuladores es una pequeña caja de color amarillo etiquetada con el nombre o el símbolo de la operación realizada.

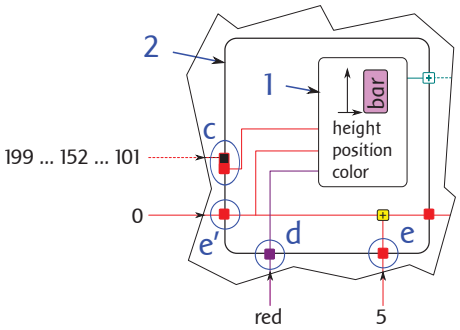


Figura 6. Extracto del patrón que representa la iteración sobre los valores de una serie.

Bucles. Pensados para operar en cada elemento de una lista o hacer tareas repetitivas. Visualmente, son grandes cuadros en blanco, lo que nos permite incluir operaciones dentro de ellos. Para pasar un flujo dentro de la caja con el fin de ser utilizado por las operaciones interiores, tenemos que utilizar los ganchos:

Ganchos. Son los huecos en un bucle por los que pueden pasar flujos, para que se puedan utilizar sus valores dentro de los mismos.

En primer lugar, los *ganchos simples* (visualmente, cuadrados individuales en la izquierda o la parte inferior de la caja) pasan el flujo recibido al bucle tal cual. Son útiles para pasar los valores que, calculados en el exterior del bucle, son comunes a todas sus iteraciones.

Yendo más lejos, podemos cambiar el valor del flujo entre iteraciones, simplemente la creación de un *gancho emparejado* en el lado derecho de la misma caja, que actúa como una entrada encadenada para la siguiente iteración. Esta es la forma más fácil de mantener un valor cambiante entre iteraciones. Además, el valor final puede ser obtenido al final del bucle.

A continuación tenemos los *ganchos dobles*, que reciben un flujo de secuencia y pasan cada elemento a las iteraciones sucesivas. Visualmente, se representan con un cuadrado doble.

Podemos secuenciar la salida de cualquier operación interna, encadenando los valores de salida iteración sobre iteración. Esto se logra mediante los *ganchos de secuencia*, visualizados como cuadrados, a la derecha del bucle, que incluyen un + signo positivo.

Vamos a continuar construyendo el modelo del gráfico de barras con los conceptos recién explicados. Reutilizando el módulo que crea una *barra*, construiremos una *serie* usando un bucle. Los requisitos son los siguientes.

c. Para cada valor numérico de la serie habrá una barra cuya altura es el valor.
d. Todas las barras de la misma serie tiene el mismo color.
e. Dentro de una serie de datos, las barras están espaciadas un número conocido de unidades.

En la Fig. 6 podemos ver el bucle (2) que permite iterar la lista de valores con un gancho *doble*. Dentro del bucle, para cada valor construimos una barra (1) invocando el patrón desarrollado recientemente como sub-módulo. Cada invocación al patrón de barras recibe `(c)` el valor iterado como entrada, a partir del gancho *doble*. La etiqueta `d` indica que el color serie se pasa a través de un gancho *simple*, ya que es el mismo para todas las barras de la misma serie.

La posición-x de las barras necesita el uso de dos ganchos. Uno de ellos es *simple* y representa la separación entre las barras de la misma serie `(e)`, ya que se puede calcular fuera del bucle. El otro es un gancho *emparejado*, y su valor comienza en un punto conocido `(e')` que representa la posición de la primera barra. Este valor es modificado dentro del bucle, iteración a iteración, para calcular la posición de cada barra. En este pequeño ejemplo podemos ver también un gancho *de secuencia* en la esquina superior derecha, que empaqueta las salidas de las invocaciones del patrón *barra*. Es decir, genera una secuencia con todas las barras creadas. Un elemento nuevo de la biblioteca nativa será finalmente explicado con el fin de ayudarnos a completar el patrón.

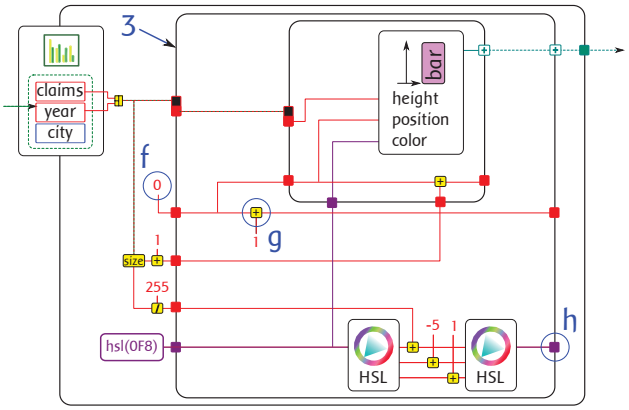


Figura 7. Implementación del gráfico de barras al completo, usando el DSL.

Correspondencias y conversiones. A veces es necesario transformar los datos de un espacio de valores conocido a otro. Varios modelos se proporcionan para describir la mayoría de los tipos nativos, ya que existen muchas descripciones posibles.

Por ejemplo, un flujo de tipo-color puede ser empaquetado (y desempaquetado), utilizando CMYK, HSL o RGB. Por otra parte, podemos construir ángulos basados en grados sexagesimales, centesimales o radianes; y posiciones que hacen uso de coordenadas cartesianas (x, y), o polares $(r\theta)$...

Podemos utilizarlas, por ejemplo, para asignar un valor con un mínimo y un máximo conocido a una escala de colores. Estas operaciones pueden generar una representación visual de la propia asignación, generando una *leyenda*.

Con este concepto podemos terminar el patrón. El paso final del desarrollo está en la Fig. 7. El bucle (3) recibe la secuencia de series como una entrada.

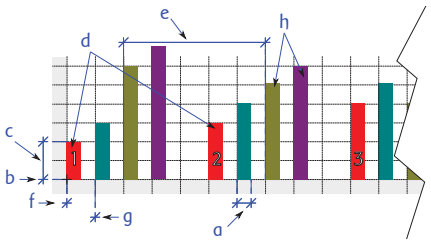


Figura 8. Geometría subyacente de la Fig. 1.

```
module "bar"
require "standard.vml"
input height: Integer,
    position: Position, color
    : Color

Builder b -> Rectangle {
  height <- input.height
  width <- 1/2
  x <- input.position
  y <- 0
  color <- input.color
}
output <- b.out
```

Figura 9. Representación del patrón-barra de la Fig. 4 haciendo uso de la sintaxis textual del DSL.

La primera barra de la primera serie comienza en $x = 0$ (`f`), por lo que hay que utilizar una constante. El espaciado entre las series (`g`) requiere que el inicio de una sea una unidad a la derecha de la anterior. Por otra parte, el color de las barras (`h`) es diferente entre las series. Para ello, descomponemos el color mediante el modelo HSL y cambiamos por separado el tono, la saturación y la luminosidad. Después, empaquetamos el color de nuevo y encadenamos en un gancho su valor para utilizarlo en la siguiente iteración.

El uso del DSL añade comprensibilidad y, aunque al verlo por primera vez podría parecer difícil, no lo es más que desarrollar visualizaciones con cualquier otro lenguaje. La complejidad aparente al mostrar un patrón con componentes nativos no corresponde a la realidad ya que la propia capacidad de reutilización puede hacer que el desarrollo escale y no se creen redundancias. Si ahora el usuario quisiera crear el patrón de la Fig. 2 (que muestra gráficos de barra sobre puntos de un mapa) no tendría que implementarlo entero, sino reutilizar las barras y construir sólo el resto. Así mismo, puede el lector entender la analogía con un lenguaje de propósito general como Java, que se muestra muy verboso en su versión estándar para casi cualquier tarea, pero que a cambio cuenta con numerosas librerías ya desarrolladas que permiten reducir el tiempo total de desarrollo.

No hemos implementado algunos detalles menores del patrón (como las leyendas o los filtros) con el fin de mantenerlo más simple. A continuación vamos a explicar la forma en que estos modelos se pueden representar en un navegador.

6. Soporte tecnológico

El objetivo de esta sección es explicar algunas de las decisiones tomadas durante el desarrollo del sistema, como las principales tecnologías empleadas para la creación del DSL, el intérprete y la creación de las visualizaciones.

6.1. Desarrollo

La gramática del lenguaje ha sido desarrollada usando notación EBNF con el apoyo de XText[1], del proyecto Eclipse. Un ejemplo textual del DSL puede verse en el listado de la Fig. 9, que muestra la representación del patrón-barra desarrollado en la Fig. 4.

Hemos elegido XText porque proporciona la generación automática y personalizable de un completo editor con coloreado de sintaxis, validación y autocompletado. Por otra parte, genera también un meta-modelo EMF[2]. Este meta-modelo es esencial, ya que no sólo representa la parte conceptual del DSL, sino que también es el punto necesario de partida para desarrollar un editor gráfico basado en GMF[3]. Las transformaciones de modelo a texto han sido desarrollados usando Xtend[4], que está totalmente integrado con XText y con el ecosistema proporcionada por Eclipse. Estas transformaciones generan código Java ejecutable que es el que realmente procesa los datos y genera los gráficos desde el servidor.

6.2. Ejecución

Los conceptos de nodo y flujo, junto con los tipos de datos nativos, entre otros, han sido implementados en una librería Java. Esta librería se encarga de planificar el trabajo de cada nodo cuando las entradas de datos están disponibles. Utiliza la entrada y salida estándares para representar la comunicación de los patrones. Sin embargo, para la comunicación con el servidor de visualizaciones es necesario utilizar una API HTTP REST. Esto hace necesario un middleware que se encargue de atender las solicitudes y transmitirlas al intérprete. El middleware elegido ha sido Apache[5]. La serialización de los datos utilizados como entradas y salidas se realiza mediante YAML que es un super-conjunto del famoso JSON. YAML permite referencias en el documento (claves externas), y otras cuestiones necesarias para la correcta serialización de grafos, por ejemplo.

6.3. Visualización

El resultado final de un patrón de visualización es una representación abstracta de la imagen. El navegador se encargará de renderizarla. Esto permite tener motores diferentes dependiendo del dispositivo o plataforma.

[1] `https://www.eclipse.org/Xtext/`
[2] `https://www.eclipse.org/emf/`
[3] `https://www.eclipse.org/modeling/gmp/`
[4] `https://www.eclipse.org/xtend/`
[5] `http://www.apache.org/`

Actualmente, hay un motor desarrollado que hace uso de JavaScript. Como se ve en la Fig. 7, el tamaño final de la imagen no se tiene en cuenta durante la construcción del mismo. Esto es porque el proceso de representación incluye un algoritmo automático de escala y centrado (hace los cálculos necesarios para mostrar el gráfico en la zona final deseada). El resultado final de este proceso es una imagen SVG[6] que puede ser incrustada en una página Web HTML5 en tiempo de ejecución. La elección de SVG se debe a que es soportado por todos los navegadores y es un estándar del W3C. El `canvas` de la API de HTML5 se ha evitado debido a las dificultades para acceder a la imagen una vez que ha sido dibujada. Por el contrario, SVG permite acceder a prácticamente cada parte de la gráfica construida desde el código para poder manipular el gráfico, lo que en el futuro permitirá modelar la interacción con el usuario.

7. Trabajos relacionados

En esta sección vamos a revisar algunos trabajos relevantes sobre el desarrollo de visualizaciones de datos.

7.1. Académicos

El problema de hacer más fácil el desarrollo de visualizaciones ha sido un objetivo académico desde que Bertin[3] publicara su taxonomía. Cabe mencionar en este sentido, el esfuerzo de Mackinlay hacia el concepto visionario de la automatización en la selección de patrones[4]; a Shneiderman por su clasificación de las visualizaciones[5]; y a Tufte, por su trabajo inspirador y valioso[6] sobre visualizaciones estáticas y técnicas de visualización. Desde que Fowler publicó su libro acerca de DSL[7], una gran cantidad de nuevos pequeños lenguajes han ido surgiendo, para cubrir los campos más variados. *Stencil*[8], por ejemplo, es una arquitectura computacional prometedora que afirma que permitirá un rápido desarrollo de visualizaciones. Hasta ahora, este enfoque no tiene sintaxis gráfica.

Nuestro DSL es versátil y válido para diferentes plataformas de visualización ya que su motor de renderización puede evolucionar junto con las tecnologías sin poner en peligro los conceptos básicos o los patrones ya desarrollados.

7.2. Tecnológicos

Web. Recientemente, JavaScript y su evolución en la Web han hecho posible una tormenta de nuevas tecnologías, muchas materializadas en forma de librería. Citaremos los más relevantes para este trabajo. *three.js*[7] es una librería JavaScript para crear escenas en 3D en el navegador. *VVVV.js*[8] es un marco de trabajo basado en la Web que ejecuta programas interactivos especificados mediante un

[6] `http://www.w3.org/Graphics/SVG/`
[7] `http://mrdoob.github.com/three.js/`
[8] `http://vvvvjs.quasipartikel.at/`

editor gráfica. *Raphaël*[9] es también una biblioteca, con la intención de hacer más fácil la creación y administración de gráficos Web vectoriales. *Protovis*[10][9] también nació a raíz de esa idea (de hacer una capa de abstracción para SVG). Sin embargo, actualmente su equipo está trabajando en *d3.js*[11][10] que está más centrado en el acceso directo a los gráficos –lo que permite una mejor interacción– en lugar de crear un nuevo nivel (de forma similar a cómo *jQuery*[12] lo hizo hace algunos años para facilitar el acceso directo a la DOM de HTML). *Unveil.js*[13][11] también es una biblioteca que crea una capa de abstracción combinando SVG con una API. Google desarrolló, con un enfoque SaaS, un producto de visualizaciones Web llamado *Google Chart Tools*[14] que puede ser embebido en cualquier página Web mediante JavaScript.

Sin embargo, estos trabajos tienen una orientación tecnológica y no se ajustan al objetivo de ser independientes de la misma o de mantener los desarrollos válidos a medida que ésta avance. Además, presentan una cierta falta de abstracción para facilitar el desarrollo a los expertos no programadores.

Escritorio. Tecnologías de escritorio también han surgido en torno a la visualización. *Processing*[15] es una de ellas, básicamente, un lenguaje para desarrollar sistemas interactivos. Tiene una versión llamada *Processing.js*[16] que compila el código para su publicación en entornos Web. *Microsoft Excel*[17] tiene características de creación de gráficos, muy limitada, pero que reflejamos aquí por su uso generalizado. *Tableau*[18] es un programa privativo que también permite que la creación de gráficos basados en datos.

Estos enfoques, a diferencia del nuestro, no permiten fácilmente al usuario cambiar de patrón, reutilizarlo entre diferentes proyectos o participar en su desarrollo fácilmente.

8. Conclusiones y trabajo futuro

La visualización de datos es un concepto clave a la hora de representar datos para la toma de decisiones y está ganando impulso en entornos de *Business Intelligence*. Sin embargo, el desarrollo de visualización de datos no es una tarea fácil. Se tiene que enfrentar a problemas en dos niveles diferentes: tecnológico y humano / social. El trabajo presentado trata de resolver ambos problemas, siguiendo un enfoque dirigido por modelos. En el documento se ha mostrado, a

[9] http://raphaeljs.com/
[10] http://mbostock.github.com/protovis/
[11] http://mbostock.github.com/d3/
[12] http://jquery.com/
[13] https://github.com/michael/unveil
[14] https://code.google.com/intl/en/apis/chart/
[15] http://processing.org/
[16] http://processingjs.org/
[17] http://office.microsoft.com/es-es/excel/
[18] http://www.tableausoftware.com/

través de un ejemplo, cuáles son los principales pilares de un DSL para el modelado visual y reutilizable de patrones. Desde un punto de vista computacional el uso del DSL favorece una potencial paralelización de tareas, ya que se basa en un paradigma *Data-flow*.

Como siguiente paso en este trabajo se va a desarrollar un editor visual basado en GMF o alguna otra tecnología que se apoye en el proyecto Eclipse. Actualmente sólo están desarrolladas las parte que cubren la sintaxis textual (editor), el metamodelo y las transformaciones a código Java. El trabajo futuro incluye además la definición de un sistema experto para sugerir la visualización más adecuada para un determinado conjunto de datos basando la decisión en metadatos y otra información social, como la forma en que estos datos han sido visualizados en el pasado por otras personas, etcétera. El desarrollo de un depurador con la posibilidad de ver los patrones trabajando con datos reales y hacer cada vez más patrones de la biblioteca estándar también se encuentra en nuestra hoja de ruta.

Referencias

1. Michael G Brooks. The Business Case for Advanced Data Visualization, 2008.
2. Rober Morales-Chaparro, Juan Carlos Preciado, and Fernando Sánchez-Figueroa. Data-driven and User-driven Multidimensional Data Visualization. In *Explore Web. Workshops of International Conference on Web Engineering.*, 2011.
3. Jacques Bertin. *Semiologie Graphique: Les Diagrammes, Les Reseaux, Les Cartes.* 1967.
4. Jock Mackinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, 5(2):110–141, April 1986.
5. Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. *Visual Languages, 1996. Proceedings., IEEE*, 1996.
6. E.R. Tufte and G. Howard. *The visual display of quantitative information*, volume 16. Graphics press Cheshire, CT, 1983.
7. Martin Fowler. *Domain Specific Languages*, volume 1 of *Programming the Memory Heirarchy.* Addison-Wesley Professional, 2010.
8. Joseph A Cottam. *Design and implementation of a stream-based visualization language.* PhD thesis, 2011.
9. Jeffrey Heer and Michael Bostock. Declarative language design for interactive visualization. *IEEE transactions on visualization and computer graphics*, 16(6):1149–56, 2010.
10. Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D^3: Data-Driven Documents. *IEEE transactions on visualization and computer graphics*, 17(12):2301–9, December 2011.
11. Michael Aufreiter. *Web-based Information Visualization.* Master's thesis, 2011.

Managing crosscutting concerns in component based systems using a model driven development approach

Pedro J. Clemente, Juan Hernández José M. Conejero, and Guadalupe Ortiz

Quercus Software Engineering Group. University of Extremadura (Spain)
{pjclemente, juanher, chemacm, gobellot}@unex.es**

1 Summary

In the last few years, *Model-Driven Development*, *Aspect-Oriented Software Development (AOSD)*, and *Component Based Software Development (CBSD)* have come to be accepted as strong options with which to tackle the design and semi-automatic construction of complex distributed applications.

In general, one may consider that the ultimate goal of these proposals is to be able to reduce development costs and effort, while improving the modularity, flexibility, adaptability, and reliability of software systems. An analysis of each of these technologies shows them all to include the principle of the separation of concerns and their further integration as key factors to obtaining high-quality and evolvable large software systems. Each identifies different concerns and deals with them separately in order to specify, design, and build applications, and at the same time provides mechanisms for the correct and appropriate integration of these concerns in the final application.

Thus, in this paper, we present a fully MDA compliant approach to developing component-and-aspect-based software systems (see figure 1). The approach allows the system to be modeled at different abstraction levels, from early platform independent models to platform specific ones and to the final code of the system. In particular, the system is obtained by automatic model transformations through four refinements based on: (i) an aspect-component UML model (figure 1-1) based on a aspect-component UML profile; (ii) a UML 2.0 component model (figure 1-2); (iii) a UML model for the CCM platform (figure 1-3) based on a UML profile for CCM; and (iv) the final code of the system for CCM (figure 1-4, figure 1-5, figure 1-6). Based on these refinements, the following conclusions may be drawn. Firstly, the aspect-component profile allows the encapsulation of crosscutting concerns in separate entities, thus removing crosscutting from the core components. Secondly, the UML 2.0 component model ensures that standard techniques of generating code from these models may be re-used, thus filling the gap between component-and-aspect-based models and component-based ones. Thirdly, the model based on the CCM profile provides

** This work was supported by Junta de Extremadura (GR10129), MEC and FEDER under contracts TIN2008-02985 and TIN2011-27340.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 579-580, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

the details related to a specific platform and can thus also benefit from standard transformations constructed for that platform. And fourthly, the automatic code generation technique used by the approach provides a way of relieving the developer from the need to deal with CCM specific programming details.

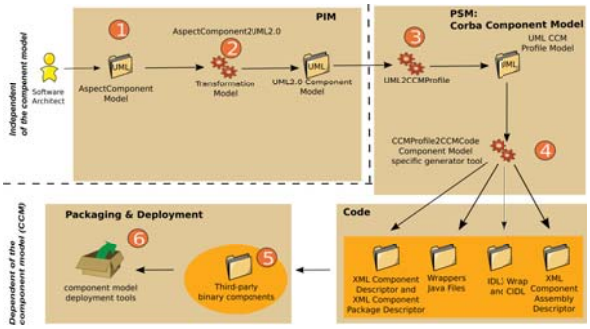


Fig. 1. Development process based on components & aspects.

Besides, by means of a twofold empirical analysis (modularity analysis at the model level and complexity analysis of component-based systems), we evaluated the possible benefits of our approach in terms of two internal quality attributes defined in ISO/IEC 9126 – modularity and complexity. These quality attributes were measured in two versions of the well-known PetStore system: the original system and an aspect-oriented version modeled with our approach. We observed that: (i) modularity is greatly improved in the AO version since crosscutting concerns are appropriately encapsulated into separate entities; (ii) the complexity of the component-based models is also reduced due to the reduction of coupling dependencies between components. Moreover, since the analysis was driven by a set of concern-oriented metrics previously validated in the literature, it was possible to infer further benefits regarding other quality attributes. In particular, since the presence of crosscutting had been empirically correlated with stability, changeability, and error-proneness, our empirical analysis also provided indications that the AO version of PetStore is more stable, easier to change, and less error-prone. Likewise, the reduction of complexity and the automatic generation of source code facilitate the components' reusability to different systems. This empirical evaluation thus lends support to the claims made above about the benefits obtained in our work.

References

1. Pedro J. Clemente, Juan Hernández, José M. Conejero and Guadalupe Ortiz. Managing crosscutting concerns in component based systems using a model driven development approach. Journal of Systems and Software. Vol 84(6), pages 1032-1053. January 2011.

Sesión Temática 6

Líneas de Producto, Componentes y Arquitecturas Software

Coordinadores: *Dr. Carlos Canal y Dr. Silvia Abrahão*

Sesión Temática 6: Líneas de Producto, Componentes y Arquitecturas Software
Coordinadores: Dr. Carlos Canal y Dr. Silvia Abrahão

Sebastián Villarroya Fernández, David Mera, Manuel A. Regueiro and José Manuel Cotos. *Diseño de Servidores de Adquisición y Publicación de Datos de Sensores*. (Regular)

Jesús García-Galán, Pablo Trinidad and Rafael Capilla. *Automating the deployment of componentized systems*. (Emergente)

Javier Cámara and Rogerio De Lemos. *Towards Run-time Resilience Evaluation in Self-Adaptive Systems*. (Emergente)

Juan F. Ingles-Romero, et al. *Prototyping component-based self-adaptive systems with Maude*. (Regular)

Francisco Sánchez-Ledesma, Juan Pastor y Diego Alonso. *Entorno de desarrollo de aplicaciones para un framework de componentes*. (Herramienta)

Jessica Díaz, Jennifer Pérez, Pedro P. Alarcón and Juan Garbajosa. *Agile Product Line Engineering—A Systematic Literature Review*. (Relevante)

Abel Gómez, Mª Carmen Penadés and José H. Canós. *Generación de Documentos con Contenido Variable en DPLfw*. (Regular)

Sergio Segura, et al. *BeTTy: Un Framework de Pruebas para el Análisis Automático de Modelos de Características*. (Herramienta)

Silvia Abrahão, Sonia Montagud and Emilio Insfran. *A Systematic Review of Quality Attributes and Measures for Software Product Lines*. (Relevante)

A. Ruíz, L. Iribarne (Eds.): Actas de las "*XVII Jornadas de Ingeniería del Software y Bases de Datos (JISBD'2012)*", Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

Diseño de Servidores de Adquisición y Publicación de Datos de Sensores *

Sebastián Villarroya, David Mera, Manuel A. Regueiro, and José M. Cotos

Grupo de Gráficos por Ordenador e Ingeniería de Datos (COGRADE),
Departamento de Electrónica y Computación,
Universidad de Santiago de Compostela
Constantino Candeira S/N, Santiago de Compostela, España
sebastian.villarroya@usc.es
david.mera@usc.es
manuelantonio.regueiro@usc.es
manel.cotos@usc.es

Resumen. En este artículo proponemos el diseño e implementación de un framework para el desarrollo de servidores de adquisición y publicación de datos llamado DADIS. La estructura de DADIS está dividida en tres capas: i) una capa inferior de adquisición de datos que se encarga de la comunicación con los diferentes sensores, ii) una capa intermedia que constituye el núcleo del sistema y se encarga de proporcionar funcionalidad de propósito general, y iii) una capa superior de comunicación con aplicaciones de usuario. El uso extensivo del patrón de diseño *Adapter* (Wrapper) convierte a DADIS en una herramienta de propósito general extremadamente flexible en la incorporación tanto de nuevos canales síncronos y asíncronos de adquisición de datos como de nuevos servicios de publicación. Además, el uso del patrón *Observer* en la capa de comunicación con las aplicaciones de usuario permite que los servicios de publicación que se quieran incorporar puedan implementar tanto el modelo *cliente/servidor* como el modelo *publicador/suscriptor*.

Palabras Clave: Adquisición de Datos, Almacenamiento de Datos, Monitorización, SCADA, Control Distribuido, Arquitectura de Sistemas.

1 Introducción

La gran cantidad de sistemas de información geográfica (SIG) que se están desarrollando en la actualidad se benefician del crecimiento exponencial experimentado en el número de pequeños dispositivos diseñados para la observación

* Este trabajo ha sido parcialmente subvencionado por el Ministerio de Educación y Ciencia, Gobierno de España (ref. PSE-370300-2006-1). El trabajo de José R.R. Viqueira ha sido parcialmente subvencionado por el Ministerio de Ciencia e Innovación, Gobierno de España (ref. TIN2010-21246-C02-02) y la Xunta de Galicia (ref. PGIDIT 09MDS034522PR)

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 583-596, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

de diferentes parámetros de nuestro entorno. A su vez, este hecho resulta beneficioso para multitud de áreas de investigación y desarrollo (Gestión de Datos Geográficos, Registro de Datos, Sistemas de Información Medioambiental, Sistemas de Adquisición y Control Supervisado (SCADA), Redes de Sensores, Sistemas de Control Distribuido) que pueden utilizar esta gran cantidad de información de forma eficiente e inteligente. Uno de los campos más activos en este sentido en los últimos años está siendo el de *Inteligencia Ambiental*.

La arquitectura general del sistema, cuando hablamos tanto de aplicaciones de Monitorización y Adquisición de Datos [8, 13] como de sistemas de Control Supervisado que no son en tiempo real [3, 5], suele estar dividida en tres capas. Cada una de ellas con una funcionalidad bien definida. En el nivel superior tenemos las *Aplicaciones de Usuario*. Estas aplicaciones le proporcionan al usuario final la funcionalidad necesaria para llevar a cabo tareas como la visualización y análisis de los datos. Dicha funcionalidad es fuertemente dependiente del dominio del problema y de las preferencias de usuario. En el nivel inferior se llevan a cabo los procesos de observación que hacen uso de *Dispositivos de Toma de Datos*. Debido a la gran variedad de estos dispositivos se presenta una gran heterogeneidad tanto en su funcionalidad como en sus capacidades de comunicación, incluso si nos restringimos a una área de aplicación concreta. Otro factor que presenta un alto grado de variabilidad es el desarrollo de software para estos dispositivos, debido a su gran dependencia de las especificaciones del fabricante. Como vemos, se hace necesaria una tecnología que por un lado permita el acceso homogéneo a todos los dispositivos de toma de datos y por otro proporcione la funcionalidad necesaria para que las aplicaciones de usuario puedan programar y planificar tanto los procesos de adquisición de datos como el almacenamiento persistente de los datos observados. Por tanto, es en el nivel intermedio donde se lleva a cabo la implementación de esta tecnología haciendo uso de *Servidores de Datos* que se comportan como pasarelas o puertas de enlace entre la parte de tecnologías de la información y comunicación y la gran cantidad de buses de campo y comunicaciones serie/paralelo utilizados por dispositivos específicos en el ámbito industrial.

Mientras la funcionalidad de las aplicaciones de usuario depende del dominio de aplicación y el desarrollo de software para dispositivos de adquisición de datos depende de las especificaciones del fabricante, la funcionalidad y arquitectura de los servidores de datos no varían demasiado de una aplicación a otra.

Este artículo presenta una generalización del diseño de un servidor de adquisición y publicación de datos. En concreto se presenta un framework llamado DADIS que permite:

- la incorporación flexible de nuevos servicios de publicación que pueden ser añadidos de forma sencilla sin que se vean afectadas otras partes del servidor.
- la incorporación flexible de nuevos dispositivos de toma de datos que proporcionen información a través de diferentes canales de comunicación.
- el almacenamiento persistente y totalmente parametrizable de las medidas.
- la incorporación de nuevos servicios de administración para llevar a cabo la administración remota de forma totalmente flexible.

Gracias a estas características podemos asegurar que DADIS facilita enormente la implementación de servidores de adquisición y publicación en aplicaciones de monitorización. Por tanto, hasta donde alcanza el conocimiento de estos autores, la mayor aportación del presente trabajo radica en que la flexibilidad y funcionalidad de la solución propuesta resulta de gran utilidad en los ámbitos científico e industrial y el esfuerzo para la generalización de servidores de adquisición y publicación de datos realizado durante el diseño de DADIS es original.

El resto del artículo está organizado de la siguiente forma, en la Sec. 2 se discuten trabajos anteriores relacionados con éste, se detalla en la Sec. 3 un caso de estudio en el campo de la monitorización energética en barcos de pesca que sirve de punto de partida para el desarrollo del framework, la arquitectura del framework se presenta en la Sec. 4, en la Sec. 5 se detallan en profundidad tanto el diseño como la implementación del framework, finaliza este trabajo la Sec. 6 con la exposición de las conclusiones y mención de las líneas de trabajo futuro.

2 Trabajos Relacionados

La utilización de redes de sensores en aplicaciones de control y monitorización ha sido objeto de numerosos trabajos de investigación y desarrollo a lo largo de la última década. En lo que resta de sección discutiremos algunas de esas propuestas utilizadas en diferentes áreas de aplicación.

Comenzaremos por el ámbito de aplicación principal de DADIS, es decir, la adquisición de datos para monitorización. En [10] se presenta una Red de Sensores Inalámbricos para la monitorización del suelo. Los nodos de la red toman una muestra de temperatura y humedad del suelo cada minuto y la almacenan localmente. Un nodo, que hace las funciones de pasarela entre la red y el exterior, recoge periódicamente los datos de los sensores y los almacena en una base de datos. La funcionalidad de esta pasarela es similar a la de los servidores en DADIS. Sin embargo, al contrario que en DADIS donde la solución está encaminada a la generalización, en [10] se adopta una solución *ad-hoc*.

Una solución más compleja es el framework específico de dominio para Redes de Sensores Corporales Inalámbricos (RSCI), llamado SPINE, que se presenta en [2]. La arquitectura de SPINE está formada por dos elementos: una colección de *Nodos Sensores* y un *Nodo Coordinador*. El nodo coordinador se encarga de recoger y analizar los datos, gestionar la red y actuar como pasarela entre los sensores y la red WAN exterior. El nodo sensor gestiona los sensores y sirve de capa de abstracción de éstos proporcionando una interfaz estándar para los controladores de sensor. Además, tiene la responsabilidad de muestrear los sensores y almacenar los datos obtenidos en búferes adecuados. La funcionalidad de este nodo es igual a la del componente de *Adquisición de Datos* de DADIS. Sin embargo, existen diferencias importantes entre SPINE y DADIS. Una de ellas es que, al contrario que DADIS, SPINE incorpora procesado de señal. Otra diferencia sustancial radica en el hecho de que los nodos sensores y el nodo coordinador de SPINE están estrechamente acoplados debido a que el framework

ha sido diseñado para desarrollar aplicaciones RSCI completas. Por el contrario, DADIS está pensado para desarrollar servidores de publicación y adquisición de datos y no está pensado para desarrollar aplicaciones completas. Por lo tanto, la comunicación entre servidores y clientes se puede reconfigurar mediante la incorporación de nuevos servicios de publicación.

En [7] se definen la arquitectura y los requisitos funcionales de un Sistema de Información de Observación Medioambiental. Dicha arquitectura está compuesta por cinco capas. La primera de ellas, *Infraestructura de Comunicación y Observación*, se corresponde con los canales de adquisición de datos de DADIS puesto que proporciona los sensores medioambientales y las comunicaciones de red. Otro componente importante de la arquitectura es el que se encarga del *Aseguramiento de la Calidad, Control de la Calidad y Procedencia de los Datos*. Actualmente esta funcionalidad está fuera del alcance de DADIS. Un tercer componente, *Almacenamiento de Datos y Metainformación*, proporciona la funcionalidad necesaria para el almacenamiento persistente tanto de datos de sensores como de metainformación. Como se asegura en [7], la mediación entre la variedad de software que soporta a los sensores y los sistemas de comunicación es sin duda un valor añadido. En DADIS se puede conseguir esta mediación con la implementación de adaptadores para diferentes canales de adquisición. En [7] también se destaca la importacia de la metainformación a la hora de descubrir, acceder e interpretar las observaciones. El modelo de datos persistente de DADIS incluye la metainformación típica para las observaciones pero actualmente no soporta ningún estándar para metadatos. El cuarto elemento de la arquitectura se encarga del *Descubrimiento y Presentación de los Datos*. En DADIS forma parte de la responsabilidad de los clientes y está, por tanto, fuera su alcance. Para finalizar, el último componente se encarga de la *Publicación e Interoperabilidad*. En DADIS esta funcionalidad es responsabilidad de los servicios de publicación de datos. Como hemos comentado, la incorporación de dichos servicios forma parte de la adaptación a una aplicación concreta. Por tanto, el uso de estándares de interpretabilidad apropiados para la publicación de datos es responsabilidad del usuario del framework.

Una solución parecida a la anterior se presenta en [9]. Este trabajo proporciona una infraestructura geoespacial distribuida para la "Sensor Web" llamada GeoSWIFT. La base de GeoSWIFT es el *Servicio de Detección* geoespacial abierto que proporciona una interfaz web para sistemas de sensores y su información geoespacial asociada. El *Servicio de Detección* de GeoSWIFT actúa como un adaptador que proporciona una interfaz estándar para que los clientes puedan recoger y acceder a las observaciones y oculta los diferentes protocolos de comunicación, estándares y formatos de datos de los sistemas de sensores. El estándar para el acceso a datos utilizado por GeoSWIFT está basado en las especificaciones de la iniciativa Sensor Web Enablement (SWE) del Open Geospatial Consortium (OGC). Las características de DADIS permiten la implementación de interfaces de servicios web OGC SWE relevantes mediante servicios de publicación de datos. Unos buenos candidatos a servicios de publicación de datos en

DADIS son los que implementan las especificaciones Sensor Observation Service (SOS) y Sensor Alert Service (SAS) del estándar OGC.

Si ampliamos el ámbito de análisis a los sistemas de adquisición y control supervisado (SCADA) [5], nos encontramos con arquitecturas organizadas en dos capas. La capa de *Servidor de Datos* se responsabiliza de la comunicación con los dispositivos de adquisición y control y almacena la información en una base de datos local. La capa de *Cliente* se encarga de la interacción con el usuario. Únicamente se consigue un acceso homogéneo a los dispositivos en los servidores de datos cuando los fabricantes proporcionan interfaces "OLE for Process Control" (OPC). DADIS proporciona un framework de propósito general que se puede utilizar para construir servidores de adquisición de datos en una arquitectura SCADA típica. La flexibilidad en la incorporación de nuevos canales de adquisición de datos en DADIS es similar a la proporcionada por OPC.

Para finalizar, abordamos el análisis de los Sistemas de Control Distribuido (DCS), que están compuestos por colecciones heterogéneas de dispositivos físicos conectados mediante multitud de protocolos de comunicación diferentes [4]. A pesar de que la funcionalidad de control de un DCS va mucho más allá de las capacidades de adquisición de datos de DADIS y de que las restricciones de tiempo real y tolerancia a fallos impuestas a los DCS no son requisitos de DADIS, podemos encontrar similitudes entre algunos componentes de algunas arquitecturas de DCS y la funcionalidad y estructura de DADIS. La capa *Bus de Campo Virtual* del framework CORFU presentado en [12] se utiliza para conseguir interoperabilidad entre el framework y cualquier tecnología de *bus de campo* comercial. Por tanto, como en DADIS, se tienen que implementar nuevos adaptadores (adaptadores de bus de campo) si se quiere adaptar el framework a canales de comunicación específicos. En el sistema TORERO [11], un dispositivo TORERO puede ser configurado y controlado remotamente desde el Entorno de Desarrollo Integrado (IDE) TORERO. La funcionalidad de control que se puede implementar en un DCS TORERO incluye las capacidades de adquisición de datos de DADIS. Sin embargo, si queremos conseguir la funcionalidad de DADIS en TORERO se deben realizar tareas de programación. Además, la capacidad de extensibilidad de DADIS no está presente en TORERO.

Si nos restringimos a la subcategoría de los DCS formada por las aplicaciones de telecontrol, cuya funcionalidad se limita a tareas de monitorización, podemos encontrar desarrollos como los expuestos en [3]. La funcionalidad de las *Unidades Remotas* de [3] es similar a la de DADIS si exceptuamos las capacidades de control de las primeras. Tanto DADIS como las *Unidades Remotas* son flexibles en la incorporación de nuevos dispositivos, sin embargo DADIS ha sido diseñado con una flexibilidad en el control remoto del servidor y en el almacenamiento y publicación de datos que no está presente en [3].

Hemos visto que varios componentes de sistemas y frameworks tienen similitudes con DADIS y hemos comprobado cómo la heterogeneidad en el acceso a protocolos y dispositivos es un problema que ha sido satisfactoriamente solventado en DADIS.

3 Caso de Estudio

La eficiencia energética es, en todos los ámbitos, una necesidad tecnológica de gran importancia tanto a nivel económico como medioambiental. No lo es menos en el sector pesquero. La experiencia es que en España no ha sido posible llevar al precio del pescado el aumento de costes de combustible, tal y como muestra la evolución de precios del gasóleo y del pescado entre 2000 y 2005 [14]. En vista de estos datos, la administración estatal, la administración autonómica gallega, el Instituto para la Diversificación y Ahorro de la Energía (IDAE) y algunas entidades privadas relacionadas con el sector pesquero lideradas por Puerto de Celeiro deciden subvencionar el proyecto "Peixe Verde".

Este proyecto nació con el objetivo claro de mejorar la eficiencia energética de las embarcaciones de pesca con el fin de reducir uno de los parámetros más significativos en los costes de explotación como es el consumo de combustible. Consecuentemente, la primera etapa del proyecto se dedicó a desarrollar un sistema de adquisición de datos de consumo y generación de energía [13] que permite obtener una radiografía energética del barco de pesca durante la realización de las faenas. Además, implementa mecanismos de monitorización y análisis de datos que permiten a los expertos de diferentes ámbitos analizar el consumo de energía en diferentes tipologías de embarcación. De esta forma pueden estudiar soluciones que mejoren la eficiencia energética.

Como podemos observar en la Fig. 1, la arquitectura de este sistema está dividida en dos subsistemas. En la parte inferior se muestra el Sistema de Adquisición de Datos (SADE) que está instalado en el barco. El elemento más importante de este sistema es el dispositivo de adquisición de datos ya que en él están instalados todos los componentes que le permiten gestionar el proceso de adquisición. Este dispositivo utiliza tres canales de comunicación para conectarse con los diferentes sensores instalados en el barco.

El protocolo NMEA-0183 implementado sobre un canal RS-232 sirve para conectar el GPS, el piloto automático, la sonda de profundidad, la estación meteorológica y los sensores de caudal de combustible.

El protocolo Modbus implementado sobre un canal RS-485 sirve para conectar los medidores de generación (alternadores) y consumo (iluminación exterior, iluminación interior, cocina, aire acondicionado, puente de mando, etc.) eléctrico.

A través de un canal Ethernet, el Sistema de Información del Buque (SIB) proporciona el estado funcional del barco (navegando, atracando, largando aparejo, etc.) y el Sistema de Estabilidad (SB) proporciona el estado operativo (cabeceo, escora, asiento, balanceo, etc.). Los sensores de torsión del eje de transmisión del barco y el router Wifi que proporciona conectividad con la red Wifi del puerto también se conectan por este canal.

El componente software principal del dispositivo de adquisición es el que se encarga de la *Adquisición y Publicación de Datos*. Podemos clasificar la funcionalidad de este componente en tres tipos. La *Adquisición de Datos* se encarga de muestrear las medidas proporcionadas por los sensores a través de los tres canales de comunicación mencionados anteriormente. Para la *Publicación de Datos* se han implementado tres servicios (componentes adicionales). La *Ex-*

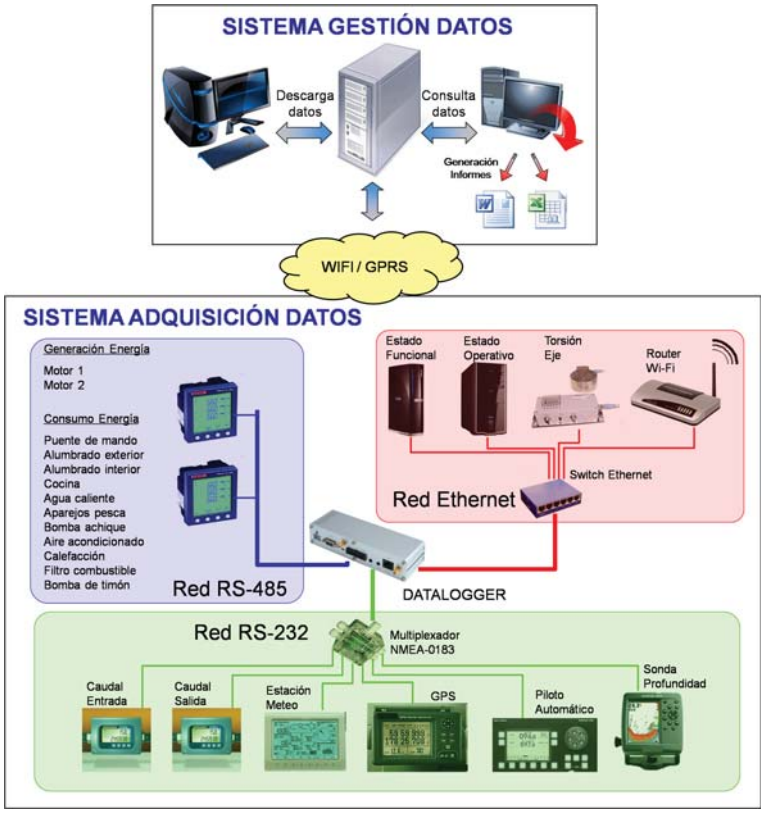


Fig. 1. Sistema de Adquisición y Monitorización del proyecto "Peixe Verde".

portación de Datos se encarga de generar y exportar ficheros de datos a partir de las medidas adquiridas. También se puede acceder a dichos ficheros a través de un *Servidor FTP* instalado en el dispositivo. En tercer lugar, el servicio de *Streaming de Datos* permite la monitorización remota del proceso de adquisición. Para completar la funcionalidad del componente de *Adquisición y Publicación de Datos*, haciendo uso del *Servidor Telnet* se puede configurar y controlar remotamente el proceso de adquisición.

En la parte superior de la Fig. 1 se muestra el Sistema de Análisis y Monitorización de Datos (ACDSADE) localizado en las instalaciones del Puerto de Celeiro. La funcionalidad de este sistema incluye tareas como monitorización remota a través del servicio de *Streaming de Datos* del componente *Adquisición y Publicación de Datos* mencionado anteriomente, análisis y consulta de datos específicos de dominio, y acceso web a los datos almacenados localmente.

La comunicación entre el SADE y el ACDSADE se realiza a través de Internet. El SADE tiene conectividad gracias a las redes WIFI y GRPS a las que tiene acceso únicamente en las inmediaciones del Puerto de Celeiro.

En las siguientes secciones se detalla la generalización del diseño del framework basado en el caso de estudio explicado en esta sección y en los requisitos de sistemas de monitorización y adquisición de datos similares desarrollados para otros dominios de aplicación. La funcionalidad del framework llamado DADIS generaliza la proporcionada por el componente software *Adquisición y Publicación de Datos* del SADE presentado en el caso de estudio anterior. La flexibilidad en la incorporación de nuevos canales de comunicación, nuevos servicios de publicación de datos y nuevas capacidades de control remoto se ha establecido como un requisito fundamental en el diseño de DADIS.

4 Arquitectura del Framework

En la Fig. 2 podemos observar las tres capas de software en las que se distribuyen los componentes que forman parte de la arquitectura del framework DADIS.

En el nivel inferior, *Adquisición de Datos*, tenemos el componente *AdquisiciónDatos* que se encarga de muestrear la información que proporcionan los sensores conectados a los diferentes canales de adquisición. La frecuencia de muestreo es configurable para cada parámetro y forma parte de la información de configuración gestionada por *GestorConfiguración*, ubicado en el nivel inmediatamente superior. Cada vez que adquiere una medida, el componente *AdquisiciónDatos* obtiene una marca temporal a través de la interfaz *iReloj* de *GestorReloj* y envía dicha marca junto con la medida a *GestorDatos* a través de la interfaz *iGestDat*.

Aunque algunos sensores utilizados puedan tener un funcionamiento *basado en eventos* (event-triggered), el proceso de adquisición en DADIS siempre está *basado en tiempo* (time-triggered). Como muestra la Fig. 2, cada canal de adquisición debe tener asociado un gestor de canal externo. En DADIS se han definido dos tipos de canales diferentes. Mientras los canales asíncronos, *GestorCanalAsinc*, acceden a sensores de tipo "event-triggered", los canales síncronos, *GestorCanalSinc*, permiten las consultas a sensores de tipo "time-triggered". Dichas consultas se envían directamente al *GestorCanalSinc* correspondiente siguiendo el modelo *cliente/servidor*. El modelo utilizado con los sensores de tipo "event-triggered" es completamente diferente. Cada vez que un sensor dispone de una medida, el *GestorCanalAsinc* correspondiente se la envía a DADIS haciendo uso de la interfaz *iInsDatAdq*. Los datos que se reciben se van almacenando en un búfer, ubicado en *AdquisiciónDatos*, del que serán muestreados a la frecuencia de muestreo configurada para ese parámetro.

En el nivel intermedio, *Gestión de Datos y Control*, están implementados los componentes que dotan a DADIS de la funcionalidad esencial para controlar el sistema y gestionar los datos, la configuración y el reloj del sistema. El componente *GestorConfiguración* gestiona toda la información de configuración del sistema. Cualquier otro componente debe hacer uso de éste para acceder a su información de configuración. Cualquier adición o modificación de dicha información debe ser realizada por un administrador de DADIS haciendo uso del componente *GestorControl*, al cual puede acceder a través del componente *ControlRemoto* ubicado en el nivel superior.

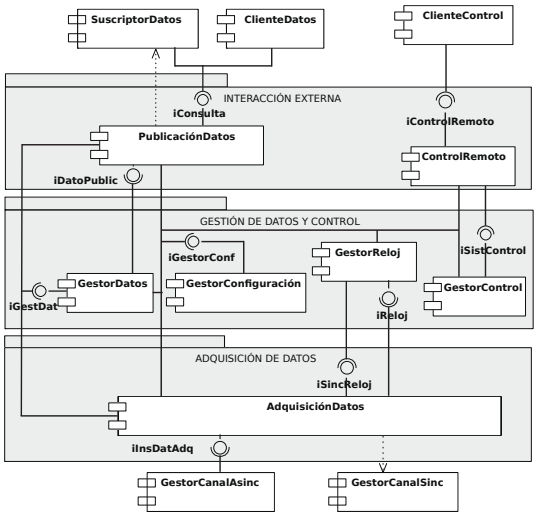


Fig. 2. Arquitectura Software de DADIS.

GestorControl no solo da acceso a la información de configuración sino que permite al administrador ejecutar tareas de control como iniciar, parar y reiniciar diferentes partes del sistema. Para aplicar cambios en la configuración de DADIS, *GestorControl* debe reiniciar los componentes del sistema que se hayan visto afectados por la modificación. Otra característica de DADIS es que existe cierta dependencia entre varios componentes que debe ser tenida en cuenta a la hora de iniciarlos o pararlos.

El componente *GestorReloj* se encarga de mantener actualizado el reloj de sistema. Cada cierto tiempo hace uso de la interfaz *iSincReloj* para obtener la fecha y hora actuales y sincronizar con ellas el reloj. Dicha información temporal es adquirida a través de un canal de adquisición determinado. Tanto el periodo de sincronización del reloj como el canal de adquisición utilizado vienen determinados en la información de configuración.

El componente *GestorDatos* permite el almacenamiento persistente de las medidas registradas por los sensores. Para conseguir esta funcionalidad, el componente *AdquisiciónDatos* hace uso de la interfaz *iGestDat*. Esta misma interfaz es utilizada por *PublicaciónDatos* para obtener las medidas almacenadas. Cada vez que se almacena una medida procedente de un canal de adquisición, *GestorDatos* se la envía a *PublicaciónDatos* a través de *iDatoPublic* para que éste la publique en los *SuscriptorDatos* correspondientes.

Finalmente, el nivel superior, *Interacción Externa* permite la interacción de DADIS con sus clientes y administradores. El componente *ControlRemoto* permite a los administradores externos, *ClienteControl*, realizar tareas de control y configuración a través de la interfaz *iControlRemoto*.

El componente *PublicaciónDatos* se encarga de la comunicación bidereccional con los servicios externos. Como podemos observar en la Fig. 2, DADIS permite dos tipos distintos de servicios, *ClienteDatos* y *SuscriptorDatos*. Ambos utilizan la misma intefaz, *iConsulta*, para acceder al sistema. Mientras los *ClienteDatos* pueden hacer consultas de datos almacenados, los *SuscriptorDatos* pueden hacer tanto consultas como suscripciones a datos almacenados. En el caso de consultas, éstas son delegadas en consultas a *iGestDat* para que *GestorDatos* devuelva los datos solicitados a *PublicaciónDatos* y éste los reenvíe al servicio correspondiente. En el caso de las suscripciones, *GestorDatos* le notifica a *PublicaciónDatos* cada vez que se almacena un dato y éste a su vez se lo notifica a todos los suscriptores de ese tipo de dato, si los hubiese. De esta forma, el suscriptor podrá realizar las consultas que considere oportunas. Resumiendo, hemos visto cómo se han implementado dos modelos de comunicación diferentes, el modelo *suscriptor/publicador* en la comunicación con los *SuscritorDatos* y el modelo *cliente/servidor* con los *ClienteDatos*.

5 Diseño e implementación del Framework

5.1 *AdquisiciónDatos*

En la Fig. 3 se muestra el diagrama de clases UML correspondiente a la estructura interna del componente *AdquisiciónDatos*. Como hemos visto, este componente utiliza la interfaz *iGestorConf* para acceder a la infomación de configuración, mientras *GestorReloj* accede a la interfaz *iSincReloj* para sincronizar el reloj de sistema. Además de las interfaces mencionadas en la sección anterior, vemos que *GestorControl* puede iniciar y parar el componente *AdquisiciónDatos* a través de la interfaz *iControlAdqDat*. Como se puede observar en la Fig. 3, las interfaces que acabamos de mencionar, *iSincReloj* e *iControlAdqDat*, están implementadas en la clase *GestorAdqDatos*.

El proceso de muestreo de los datos se lleva a cabo en un hilo de tipo *MuestreoParam*. *GestorAdqDatos* tendrá un hilo por cada parámetro que esté muestreando. En la figura podemos ver el pseudocódigo de este proceso. En primer lugar, el hilo se duerme la cantidad de tiempo (*samplingInterval*) indicada en la información de configuración de ese parámetro concreto. Una vez que se despierta, obtiene la siguiente muestra del parámetro haciendo uso de su canal de adquisición de datos (clase *CanalDatosAbstracto*). A continuación le añade a la muestra la información temporal obtenida a través de la interfaz *iReloj* de *GestorReloj*. Finalmente envía la muestra con información temporal a *GestorDatos* mediante su interfaz *iGestDat*.

Un *CanalDatosAbstracto* puede ser bien un *CanalDatosSinc* o bien un *CanalDatosAsinc*. Mientras el primero proporciona acceso a los canales de adquisición de datos síncronos, el segundo lo proporciona a los canales asíncronos de adquisición. *CanalDatosAsinc* va obteniendo las medidas de un búfer de datos, *BuferEntradaAsinc*, que almacena la última medida de cada parámetro y es rellenado por *GestorCanalAsinc* a través de su interfaz *iInsDatAdq*.

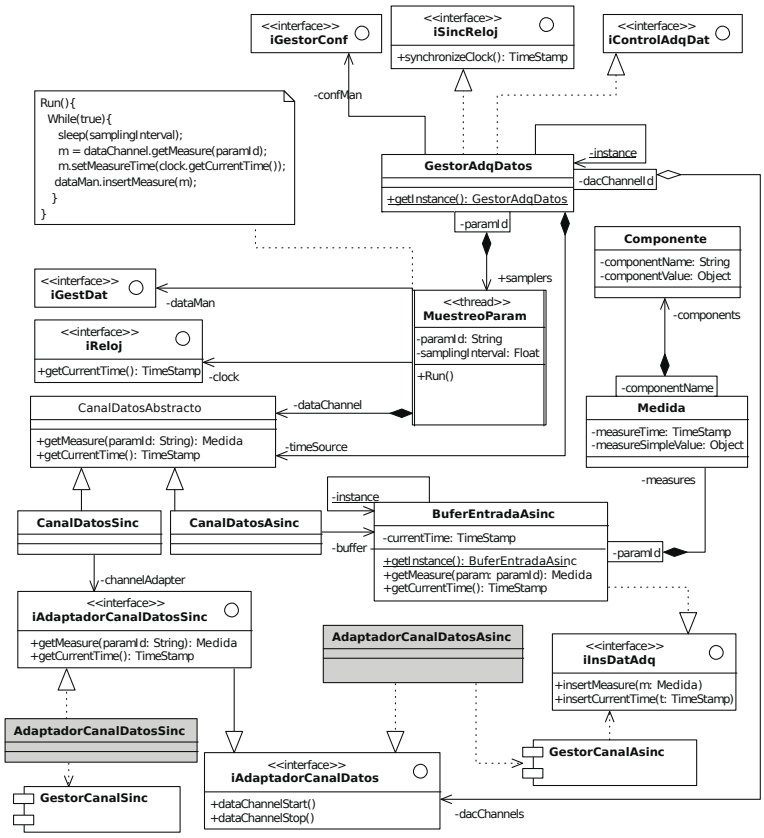


Fig. 3. Diagrama UML de *AdquisiciónDatos*.

Para que DADIS pueda acceder a la funcionalidad proporcionada por los gestores de canal de adquisición es necesaria la incorporación de clases adaptadoras específicas. Como se puede observar en la Fig. 3, por cada *GestorCanalSinc* hay que añadir una clase *AdaptadorCanalDatosSinc* que implemente la interfaz *iAdaptadorCanalDatosSinc*. De igual forma, hay que añadir una clase *AdaptadorCanalDatosAsinc* que implemente la interfaz *iAdaptadorCanalDatosAsinc* por cada *GestorCanalAsinc*. Los nombres de estas clases forman parte de la información de configuración de cada canal. Además, una lista con todos los nombres de adaptadores disponibles es gestionada por *GestorAdqDatos* para permitir que DADIS realice tareas de control sobre ellos, como apagarlos o encenderlos.

5.2 *PublicaciónDatos*

La arquitectura de *PublicaciónDatos* es muy similar a la arquitectura de *AdquisiciónDatos* presentada en 5.1 puesto que ambos componentes se encargan de la

comunicación con elementos externos. En este caso, la arquitectura consta de una sola clase, *GestorServiciosDatos*, que implementa las interfaces de control, consulta y publicación necesarias para poner su funcionalidad a disposición de los servicios externos y del resto de componentes de DADIS.

La interfaz de *control* permite al componente *GestorControl* iniciar y parar los servicios de datos registrados en una lista gestionada por *GestorServicios-Datos*. La interfaz de *consulta* permite a los suscriptores y clientes de datos externos realizar consultas sobre la base de datos del sistema. Esta interfaz consta de dos operaciones, una que permite obtener la lista de parámetros medidos y otra que permite obtener las medidas de un determinado parámetro. La interfaz de *publicación* permite al componente *GestorDatos* enviar cada medida insertada en la base de datos al suscriptor correspondiente. Para ello, *GestorServicios-Datos* almacena una lista con los suscriptores que deben ser notificados cada vez que se inserta una nueva medida de cada parámetro.

5.3 *GestorDatos* y *GestorConfiguración*

El componente *GestorConfiguración* implementa las interfaces que permiten consultar y actualizar la información de configuración así como iniciar, parar y reiniciar el componente. Los accesos a los datos de configuración de este componente consisten en una lista de servicios disponibles, una lista de canales de adquisición de datos disponibles y una lista de parámetros disponibles medidos por los sensores. Cada servicio de datos puede ser *SuscriptorDatos* o *ClienteDatos*. El sistema almacena una lista de parámetros que se utiliza para determinar si un suscriptor tiene que ser notificado o no cada vez que se inserta una medida de un determinado parámetro. Cada servicio de datos almacena el nombre del adaptador que será utilizado por *PublicaciónDatos* para interactuar con el servicio de datos externo. De igual forma, cada canal de adquisición de datos almacena el nombre del adaptador utilizado por *AdquisiciónDatos* para comunicarse con el canal de adquisición externo.

El componente *GestorDatos* implementa las interfaces que permiten iniciar, parar y reiniciar el componente. También permite consultar e insertar medidas en el almacenamiento persistente. La operación de consulta permite obtener un determinado parámetro referenciado por su identificador durante un filtro temporal dado [1]. Las medidas pueden ser simples (formadas por un valor medido y una marca temporal) o compuestas (contruidas a partir de una colección de componentes). La operación de inserción permite la inserción de una determinada medida en el almacenamiento persistente.

Para finalizar esta sección discutimos brevemente tres conocidos patrones de diseño [6] utilizados en los componentes anteriores.

El patrón *Singleton* se utiliza para restringir la instanciación de una clase a un solo elemento. La clase proporciona un único punto de acceso al objeto. Este patrón se utiliza en DADIS cuando es necesaria la coordinación en el acceso a la funcionalidad y recursos de un determinado componente. Por ejemplo, se ha utilizado este patrón en *GestorConfiguración* para coordinar el acceso a la información de configuración que está en memoria. En *AdquisiciónDatos* se

utiliza para coordinar tanto el control de los canales de adquisición y procesos de muestreo como el acceso concurrente de los diferentes *GestorCanalAsinc* y *MuestreoParam*. En *PublicaciónDatos* se utiliza este patrón en la implementación de la clase *GestorServiciosDatos* puesto que a sus recursos (servicios de datos) acceden concurrentemente varios hilos.

El patrón *Adapter* (wrapper) se utiliza para traducir de una interfaz a otra. De esta forma se pueden comunicar clases con interfaces diferentes. Este patrón es utilizado por el componente *CanalDatosSinc* de *AdquisiciónDatos* para obtener medidas directamente del componente externo *GestorCanalSinc* correspondiente. *PublicaciónDatos* lo utiliza exactamente de igual forma para acceder a los servicios de datos.

El patrón *Observer* (publicador/suscriptor) se utiliza para notificar a una lista de objetos (observadores) cualquier cambio en el estado de otro objeto concreto (sujeto). Este patrón se utiliza en la implementación de *PublicaciónDatos* para notificar a los componentes *SuscriptorDatos* cada vez que se registra una medida relevante en *GestorDatos*.

6 Conclusiones y Trabajo Futuro

El diseño de DADIS se ha descrito como una generalización para el desarrollo de servidores de adquisición y publicación de datos. Estos servidores se utilizan habitualmente en aplicaciones de monitorización porque cubren el hueco existente entre los estándares industriales usados a nivel de sensor y los estándares ICT usados a nivel de aplicación [8]. El framework soluciona el problema de heterogeneidad tanto en el accesso a datos de sensores como en las comunicaciones con las aplicaciones de monitorización y análisis. La ventaja principal del framework es que ha sido diseñado como una herramienta completamente flexible tanto en la incorporación de nuevos sensores y canales de comunicación como en la generación de servicios de datos que siguen los modelos cliente/servidor y publicador/suscriptor.

Se han utilizado diversos patrones de diseño en DADIS. El patrón singleton en el acceso concurrente a los recursos proporcionados por DADIS. El patrón observer en la implementación de servicios de datos publicador/suscriptor. Y el patrón adapter en la incorporación de nuevos servicios de datos, servicios de control remoto y canales de adquisición de datos.

Como trabajo futuro en DADIS se plantea la posibilidad de realizar experimentos que validen los beneficios de esta solución en diferentes proyectos. De esta forma se podrán contrastar los resultados obtenidos por DADIS con los resultados obtenidos por otras soluciones. Además se plantea el soporte a diversos servicios estándares propuestos por el OGC en su iniciativa SWE. En una primera etapa se propone la implementación de servicios de acceso a datos como SOS y SAS. Para ello será necesario aumentar las capacidades de las consultas de propósito general, incorporar metainformación para sensores y parámetros, e implementar el concepto "Offering" (base de datos propuesta por el OGC). En una segunda etapa se propone la implementación del Sensor Planning Service.

References

1. Allen, J.F.: Maintaining knowledge about temporal intervals. Communications of the ACM. 26, 832–843 (1983)
2. Bellifemine, F., Fortino, G., Giannantonio, R., Gravina, R., Guerrieri, A., Sgroi, M.: Spine: a domain-specific framework for rapid prototyping of wbsn applications. Software: Practice and Experience. 41, 237–265 (2011)
3. Chimaris, A.N., Papadopoulos, G.A.: Implementing a generic component-based framework for telecontrol applications. Software: Practice and Experience. 37, 1087–1132 (2007)
4. Colnaric, M., Verber, D., Halang, W.: Distributed Embedded Control Systems. Improving Dependability with Coherent Design. Springer, London (2008)
5. Daneels, A., Salter, W.: What is scada?. In: Proceedings International Conference on Accelerator and Large Experimental Physics Control Systems, pp. 339–343. Trieste, Italy (1999)
6. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, NJ (1995)
7. Horsburgh, J.S., Tarboton, D.G., Maidment, D.R., Zaslavsky, I.: Components of an environmental observatory information system. Computers & Geosciences. 37, 207–218 (2011)
8. Kolar, H., Cronin, J., Hartswick, P., Sanderson, A., Bonner, J., Hotaling, L., Ambrosio, R., Liu, Z., Passow, M., Reath, M.: Complex real-time environmental monitoring of the hudson river and estuary system. IBM Journal of Research and Development. 53, 4:1–4:10 (2009)
9. Liang, S.H., Croitoru, A., Tao, C.V.: A distributed geospatial infrastructure for sensor web. Computers & Geosciences. 31, 221–231 (2005)
10. Musaloiu-E., R., Terzis, A., Szlavecz, K., Szalay, A., Cogan, J., Gray, J.: Life under your feet: A wireless soil ecology sensor network. In: Third Workshop on Embedded Networked Sensors (EmNets). Cambridge, USA (2006) `http://www.eecs.harvard.edu/emnets/` (leído el 20 de Marzo de 2012)
11. Schwab, C., Tangermann, M., Ferrarini, L.: Web based methodology for engineering and maintenance of distributed control systems: the torero approach. In: 3rd IEEE International Conference on Industrial Informatics INDIN 2005, pp. 32–37. Perth, Australia (2005)
12. Thramboulidis, K., Tranoris, C.: An architecture for the development of function block oriented engineering support systems. In: IEEE International Symposium on Computational Intelligence in Robotics and Automation, pp. 536–542. Banff, Alberta, Canada (2001)
13. Villarroya, S., Otero, M., Romero, L., Cotos, J., Pita, V.: Modular and scalable multi-interface data acquisition architecture design for energy monitoring in fishing vessels. In: Omatu, S., Rocha, M., Bravo, J.,Fernández, F., Corchado, E., Bustillo, A., Corchado, J. (eds.) Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living. LNCS, vol. 5518, pp. 531–538. Springer, Heidelberg (2009)
14. The Impact of the Increase of the Oil Prize in European Fisheries. Draft final report of the study for the Committee on fisheries of the European Parliament. Project No. IP/B/PECH/ST/2005-142. Netherlands (2006)

Automating the deployment of componentized systems *

Jesús García-Galán[1], Pablo Trinidad[1], and Rafael Capilla[2]

[1] Universidad de Sevilla
[2] Universidad Rey Juan Carlos

Abstract. Embedded and self-adaptive systems demand continuous adaptation and reconfiguration activities based on changing quality conditions and context information. As a consequence, systems have to be (re)deployed several times and software components need to be mapped onto new or existing hardware pieces. Today, the way to determine an optimal deployment in complex systems, often performed at runtime, constitutes a well-known challenge. In this paper we highlight the major problems of automatic deployment and present a research plan to reach for an UML-based solution for the deployment of componentized systems. As a first step towards a solution, we use the UML superstructure to suggest a way to redeploy UML component diagrams based on the inputs and outputs required to enact an automatic deployment process.

1 Problem context

Software systems need to be deployed and redeployed several times as the environment and context conditions often change. This is particularly important during runtime when a system already deployed changes its current configuration and it has to be redeployed (i.e.: post-deployment reconfiguration). In the era of post-deployment, modern desktop software, mobile applications, autonomic systems, and service-based systems among others, demand continuous changes in deployment activities. Furthermore, the decision to realize an optimal deployment is still challenging. Recent proposals [2, 10] attempted to automate this process based on quality concerns, such as reliability and performance.

Today, complex systems demand automatic deployment capabilities in order to keep the system updated. For instance, pervasive software is often deployed and reconfigured over dozens of embedded devices, or cloud-based systems (e.g.: Software-as-a-Service solutions) demand continuous reconfiguration activities to satisfy new customer's needs and quality concerns (e.g.: workload is moved between servers during system upgrading). Hence, the way to achieve an optimal deployment becomes a major goal. However, there is still a lack of generic approaches that model, from the architecture point of view, the inputs and the outputs used to deploy software automatically.

* This work has been partially supported by the European Commission (FEDER) and Spanish Government under CICYT project SETI (TIN2009-07366) and by the Andalusian Government under ISABEL (TIC-2533) and THEOS (TIC-5906) projects.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 597-602, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

2 Automatic deployment of component-based systems

Deployment activities need to know which software components will map into the physical model. From our view, we understand this process as a continuous activity where the output of a deployment solution feeds again the process for a new deployment (i.e.: re-deployment).

On the search of a process to automatically deploy a new system configuration, we propose a research agenda that encompasses the following three tasks: (i) Define the inputs and outputs required by a deployment process, (ii) Specify a way to map automatically software components onto hardware nodes, and (iii) Use/build a tool to automate the deployment of componentized systems as a proof of concept to check the feasibility of the solution.

In this paper we focus on the first task. In order to represent the inputs and the outputs for automatic deployment, we will use the following three models to describe the components and behavior of any software system: (i) a model to describe the software, (ii) a model to describe the hardware, and (iii) a model to describe the mappings between the software and hardware models.

From the software architecture point of view, we rely on component-based architectures and systems to modularize and do the required mappings between software and hardware. However, because every context is different, we need to describe which software and hardware constraints and dependencies are needed by a particular deployment solution.

For instance, in the physical architecture we define the non-functional properties of the hardware nodes and runtime environment on which the software runs. These physical properties of the nodes (e.g.: RAM capacity, CPU speed, etc.) are used as constraints for the software components that demand a particular system configuration, as they will drive the final software-hardware configuration. Therefore, our proposed model requires enough expressivity to support all the information required by the mappings between software modules and hardware nodes.

Sample scenario: In figure 1 we describe a deployment scenario to motivate our proposal. It consists of a cloud-based platform where two hardware devices (i.e.: one public from Google and one private using three instances of the AppScale platform) are used to deploy the software. The devices and runtime environment exhibit different non-functional properties. The software to be deployed belongs to transportation management system that uses an API, a client, and a back-end management system. The target system defines the set of constraints over the deployment platform that might change and evolve accordingly to new requirements.

In order to perform an automatic mapping for a given deployment configuration, we model the inputs and the outputs of the process using UML [11] for the following reasons:

1. UML is a standard and widely used notation to describe software systems and supported by many modeling tools because it can describe the architecture of a system from different points of view.

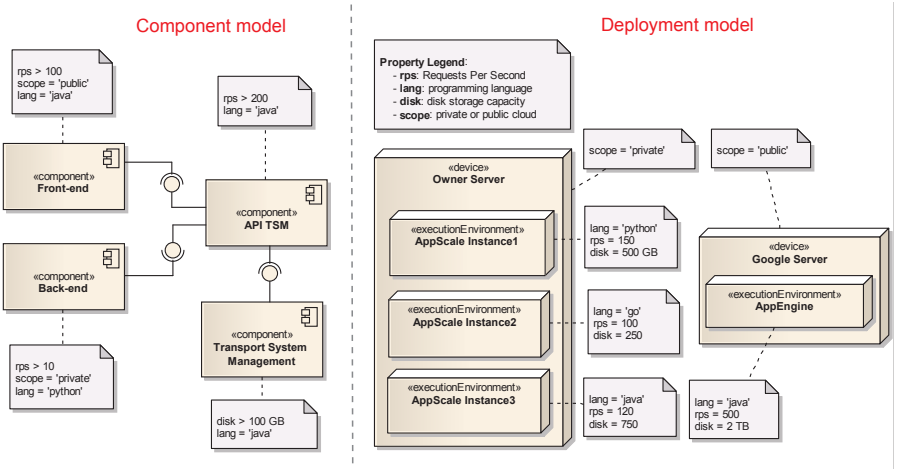


Fig. 1. A Cloud computing deployment scenario for a transportation system

2. It provides a superstructure able to support the mappings between component and deployment diagrams.

3 UML for automatic deployment inputs and outputs

Component and deployment diagrams describing the soft and hard parts of a system can be used to model the inputs and outputs of an automatic deployment process using UML. As a result, an enriched UML deployment diagram shows the allocation of software modules in hardware nodes and according to a set of hardware and software constraints defined in mapping rules.

Components use required and provided interfaces to communicate each other. For instance, service-based systems and cloud platforms often vary the deployment of the running software because changing quality conditions, when the system needs an update, or when existing hardware capabilities have to be extended. Clients that need to rebind to new services dynamically, the reallocation of services in different servers, and new cloud hardware that require to move software from one server to another, are examples of re-deployment scenarios. The component diagram of Figure 1 shows how the *API TSM* component offers two interfaces, the *Front-end* and the *Back-end* that must be allocated in hardware nodes.

In addition, UML deployment diagrams describe a high-level organization of the physical nodes (e.g.: devices, servers, etc.), and according to a particular execution environment and distribution of software modules. The allocation of software elements in the physical architecture rely on mapping rules that are used to automate the allocation process. The right side of Figure 1 displays a deployment diagram with devices belonging to two different execution environments.

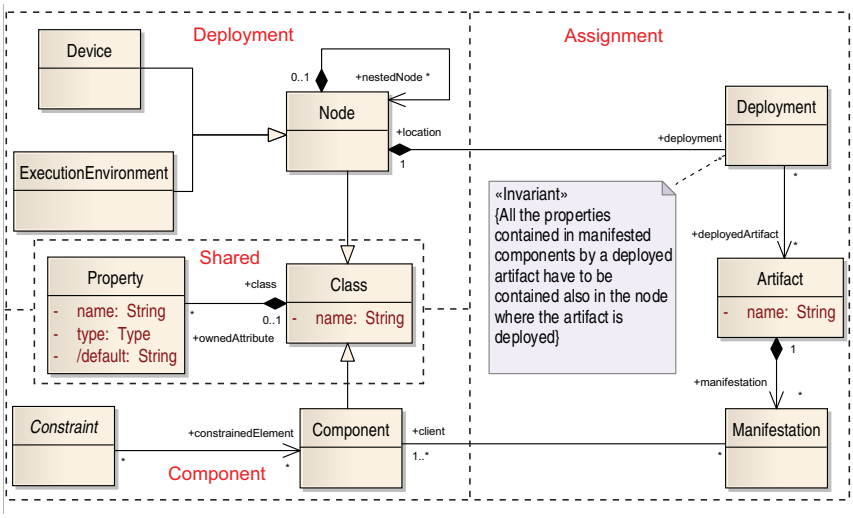


Fig. 2. Deployment with assignment metamodel based on UML superstructure

3.1 The UML Superstructure

The OMG UML Superstructure 2.4 [11] describes the formal specification of UML elements, but in most cases we use only a subset of it when designing with UML elements. In this work we identify the minimum set of elements in the UML supertructure that are needed to map software modules into hardware nodes. These elements can be also extended to provide additional capabilities for automatic deployment. As aconsequence, this superstructure provides a standard way to enhance UML, as we avoid using other approaches (e.g.: such as model-driven engineering) to perform the automatic mappings between UML models.

Figure 2 shows a simplified version of the UML superstructure containing both component and deployment diagrams. The boxes in the *Deployment* area belong to those elements required to define UML nodes in the physical view, whereas the *Component* area contains the elements to design the logical view. Both areas share the properties of the nodes which are used as constraints for the components, such as the *Shared* area displays. Finally, the *Assignment* area encompasses the elements to perform the mappings between the logical and the physical views.

In order to highlight the properties of the nodes, we use the `Property` element to exploit the inheritance relationship between the `Node` and the `Class`. Each property has a name, a type and a default value (e.g.: in Figure 1 we have examples of these properties with their allowed values, such as: rps, lang, disk or scope).

Also, the components model uses the `Constraint` class to represent the restrictions of the nodes (i.e.: properties) and it can be associated to any UML element. These constraints refer to properties that must be defined initially in

the components for deployment purposes. The component model of Figure 1 shows an example of constraints that have impact in the deployment diagram.

In the *Assignment* area, hardware elements are described by means of the `Artifact` class which defines a 1-* correspondence with software components by means of the `Manifestation` class. The deployment of software artifacts in a node is described using a `Deployment` class. In addition, we define an invariant over the Deployment in order to ensure that the components will be deployed only in those nodes that contain the properties that restrict each physical node.

4 Related work

Kruchten [6] pioneered the correspondences between architecture views early in 1995.Other authors like Clements et al. [3] modernize Kruchten's approach to define deployment viewtypes in order to allocate software modules into runtime (additional architecture views can be found in Rozanski and Woods [12]).

Regarding automatic deployment, Kramer and Magee [5] motivate this for autonomic systems while Arshad et al. [1] focus on dynamic reconfiguration using AI planning. Other related works focus on the impact of quality factors for deployment, such as Bushehrian [2] and White et al. [14] which uses performance to compute the nearest optimal deployment using simulation and evolutionary algorithms respectively. In addition, Meedeniya et al. [10] focus on reliability, while Wada et al. [13] focus on SLAs to calculate optimal deployments. Recent approches show examples of automatic deployment using an autonomous engine for service-based systems in the banking domain and using SAT solvers to optimize the best deployment configuration (Cuadrado et al. [4]), while other approaches (Malek et al. [9]) focus on deployment and redeployment activities allocating software components to hardware nodes using a trade-off of QoS properties in order to quantify the quality of the system deployment. All these approaches use proprietary notations or focus on optimization techniques, but none of them exploit the use of the UML superstructure for establishing the correspondence between component and deployment views. Only the MARTE approach (Liehr et al. [7]) exploits the use of UML profiles to allocate models automatically but focused only on UML activity diagrams for real-time execution (RTE) environments.

5 Conclusions and Future work

In this paper we present an attempt of using UML to model the inputs and outputs of an automatic deployment system. This is a first step towards a more ambitious goal of building such system. Next steps lead our future work to focus on: (i) rules to define the mapping process, and (ii) a prototype tool to demonstrate our ideas as a proof of concept using the UML superstructure. Furthermore, we believe that constraint programming can be used to compute the best deployment and model this as an optimization problem.

We are conscious that new requirements might arise in future research that conduct to changes in our proposal of UML-based inputs and outputs. To reduce

the impact of future requirements and in order to increase the confidence on the results presented in this paper, we have built a first version of a prototype research tool. From this validation, some challenging questions have appeared, such as: defining priority levels for hardware constraints, using different optimization criteria, or updating both hardware and software properties after deployment. Our prototype has been built using an MDE approach and currently, it computes basic deployment configurations using constraint programming [8]. We will focus our future efforts in dealing with these questions and tackling next steps.

Bibliography

[1] N. Arshad, D. Heimbigner, and A. L. Wolf. Deployment and dynamic reconfiguration planning for distributed software systems. *Software Quality Journal*, 2007.

[2] O. Bushehrian. Automatic object deployment for software performance enhancement. *IET Software*, 2011.

[3] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, and J. Stafford. *Documenting Software Architectures: Views and Beyond*. Addison-Wesley Professional, 2002.

[4] F. Cuadrado, J. Duenas, and R. Garcia-Carmona. An autonomous engine for services configuration and deployment. *Software Engineering, IEEE Transactions on*, 2012.

[5] J. Kramer and J. Magee. Self-managed systems: an architectural challenge. *Computing*, 2007.

[6] P. Kruchten. The 4+ 1 view model of architecture. *Software, IEEE*, 1995.

[7] A. W. Liehr, H. S. Rolfs, K. Buchenrieder, and U. Nageldinger. Generating marte allocation models from activity threads. In *FDL*, pages 215–220, 2008.

[8] A. Mackworth. Consistency in Networks of Relations. *Artificial Intelligence*, 1977.

[9] S. Malek, N. Medvidovic, and M. Mikic-Rakic. An extensible framework for improving a distributed software system's deployment architecture. *Software Engineering, IEEE Transactions on*, 2012.

[10] I. Meedeniya, B. Buhnova, A. Aleti, and L. Grunske. Reliability-driven deployment optimization for embedded systems. *Journal of Systems and Software*, 2011.

[11] Object Management Group. *UML Superstructure specification*. 2011.

[12] N. Rozanski and E. Woods. *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives*. Addison-Wesley Professional, 2005.

[13] H. Wada, J. Suzuki, Y. Yamano, and K. Oba. Evolutionary deployment optimization for service-oriented clouds. *Software: Practice and Experience*, 2011.

[14] J. White, B. Dougherty, C. Thompson, and D. C. Schmidt. ScatterD : Spatial Deployment Optimization with Hybrid Heuristic / Evolutionary Algorithms. *ACM Transactions on Autonomous and Adaptive Systems*, 2011.

Towards Run-time Resilience Evaluation in Self-Adaptive Systems

Javier Cámara[1] and Rogério de Lemos[2]

[1] University of Coimbra, Portugal
jcmoreno@dei.uc.pt
[2] University of Kent, UK
r.delemos@kent.ac.uk

Abstract. The provision of assurances for self-adaptive systems presents its challenges, since uncertainties associated with their operating environments often hamper the provision of absolute guarantees that system properties can be satisfied. In previous work, we defined a development-time approach for the evaluation of self-adaptive systems that relies on stimulation and probabilistic model-checking to provide levels of confidence regarding service delivery. However, development-time evaluation has limitations due to the dynamic nature of self-adaptive systems, whose behavior depends on run-time conditions that continuously change. In this paper, we discuss the challenges and issues posed by the shift of resilience evaluation from development-time to run-time.

Keywords: resilience evaluation; self-adaptation; assurances; architecture model

1 Introduction

Despite recent advances in self-adaptive systems, a key aspect that still remains a challenge is the provision of assurances, that is, the collection, structuring and combination of evidence that a system satisfies a set of stated functional and non-functional properties during its operation. The main reason for this is the high degree of uncertainty associated with changes that may occur to the system itself, its environment or its goals [3]. In particular, since the behavior of the environment cannot be predicted nor controlled by the system (*e.g.*, load, network conditions, etc.), this prevents obtaining absolute guarantees that system properties can be satisfied. Moreover, unlike in consolidated model-based verification and validation techniques, models in self-adaptive systems cannot be assumed to be fixed since changes that might affect the system, its environment or its goals might also affect the models.

This intrinsic uncertainty associated with self-adaptive systems has established the need to devise new approaches for assessing whether a set of stated properties are satisfied by the system during operation while changes occur. A major requirement for these approaches is that they need to be able to provide levels of confidence, instead of establishing absolute guarantees about property satisfaction. But for that, there is the need to determine what are the limits of the system when facing changes while still providing a service that can justifiably be trusted (*i.e.*, the avoidance of failures that are unacceptably frequent or severe) [10].

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 603-608, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

In a previous paper [2], we proposed an approach based on system environment stimulation and probabilistic model-checking, for the evaluation of self-adaptive systems whose environment exhibits stochastic behavior [7]. This approach enables reasoning in terms of probabilities and quantification of levels of confidence regarding the satisfaction of system properties. However, that approach has some limitations derived mainly from the fact that property verification is carried out at development-time, and feedback regarding probabilities cannot be exploited by the self-adaptive system, which is inherently of a dynamic nature. This paper discusses challenges posed by the shift of probabilistic evaluation of resilience properties in self-adaptive systems from development-time to run-time, proposing new ideas to overcome current limitations and integrating the process into the closed control loop characteristic of self-adaptive systems.

The rest of this paper is organised as follows. Section 2 introduces some background regarding the model of self-adaptive system and the kind of properties that we deal with in our approach. Section 3 overviews our approach to probabilistic resilience evaluation for self-adaptive systems. Section 4 discusses the limitations of the existing approach and outlines a run-time approach to overcome them. Finally, Section 5 outlines future research directions.

2 Background

In this section, we introduce some key concepts in relation to the evaluation of resilience in self-adaptive systems (sometimes referred to as autonomic, self-managed, or self-healing), in the context of the MAPE-K control loop [8]. Concretely, we focus on the kind of systems that exploit architectural models as part of their knowledge base for reasoning about the system under management [5,11].

Operational Profiles. Within a self-adaptive system, we distinguish between a *conventional operational profile* in which the system is operating without experiencing any anomalies, and *non-conventional operational profiles* associated with changes in the environment that induce anomalies in the system (typically triggering *adaptations*). Therefore, an adaptation is triggered as a response to a system anomaly caused by a change in the environment of the system. When the system experiences an anomaly, it goes from its conventional operational profile into a non-conventional operational profile. As a general rule, the intended purpose of the adaptation is steering the system back to its conventional operational profile.

Adaptations. We define adaptations in terms of: (i) applicability conditions, (ii) intended effects on the system, and (iii) expected deadline to fulfill its intended effect. Hence, we abstract away from the specific actions that a particular adaptation carries out. An adaptation is always related to a non-conventional operational profile through its applicability condition (*i.e.*, typically, an anomaly that triggers the adaptation).

Non-Conventional Operational Profile Models. A system model consists of a set of non-conventional operational profile models, each of which is associated with a set of adaptations through their applicability condition. To model the probabilistic behaviour

of our system we employ Discrete-Time Markov Chains (DTMCs) [9], defined as state-transition systems augmented with probabilities. DTMCs are discrete stochastic processes where the probability distribution of future states depend only upon the current state. Concretely, initial states in the models correspond to those in which the system enters the non-conventional operational profile, and the state space of each of the models includes all the states reachable from initial states, before the maximum of the deadlines for adaptations expires.

Resilience Properties. To express resilience properties, we use Probabilistic Computation Tree Logic (PCTL) [1]. PCTL provides a probabilistic operator that enables the quantification of the probability of satisfaction of an PCTL path formula. In particular, we can verify properties that represent the response (effect of adaptation) to a particular change (anomaly that triggers it) in the environment of the system. These properties include a probability bound and a time bound, and are instanced by using probabilistic response patterns [6]. With these patterns, we can express properties such as: *"When response time in the system goes above threshold* `MAX_RSPTIME`*, the probability of lowering response time below* `MAX_RSPTIME` *in 120 seconds is greater than 0.85"*.

3 Resilience Evaluation in Self-Adaptive Systems

In a nutshell, evaluating the kind of resilience properties described in the previous section in a self-adaptive system consists in stimulating its environment in a controlled way to exercise its adaptive capabilities, and to collect data about how the system reacts to those environmental changes. The collected data is aggregated into a probabilistic model of the system's behavior that will be used as input into a model checker for evaluating whether system properties are satisfied within certain confidence levels. Our approach consists of four basic steps (Figure 1) [2]:

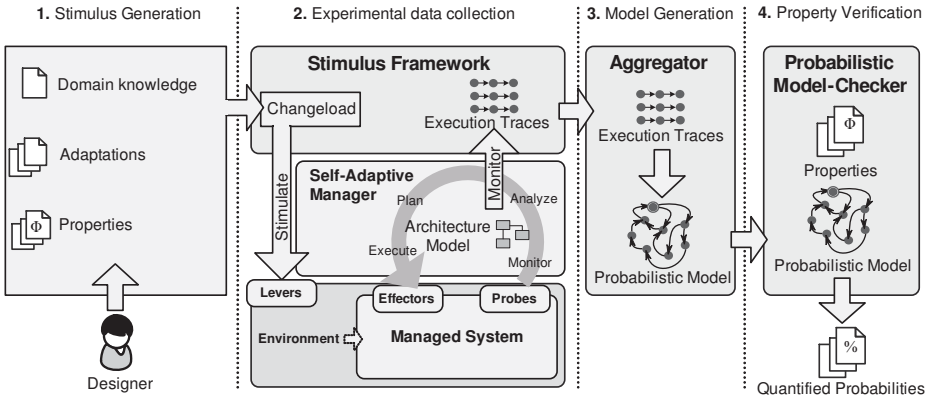


Fig. 1. Overview of resilience evaluation.

1. Stimulus generation. Determines how the system's environment should be stimulated to collect information relevant for the properties of interest. Stimulation is based on system adaptation alternatives, domain knowledge, and the properties to be verified. To obtain a representative operational model of the system, we need to subject its environment to equally representative changes. A *changeload* is a collection of scenarios comprising representative changes of the environment that stimulate the system's adaptive capabilities, enabling us to observe its response.
2. Experimental data collection. Stimulates the system's environment according to the changeload. The environmental stimulation is performed using effectors (*i.e.. levers*) to trigger adaptation in the system for collecting information, in the form of traces, about its behavior while undergoing adaptation. The information collected corresponds to time intervals that start when an anomaly occurs, and end when the deadline to fulfill adaptation goals expires. Information contained in the traces is mapped from properties of interest, which is part of the architectural model of the managed system, which is kept by the self-adaptive manager.
3. Model generation. Aggregates execution traces for each non-conventional operational profile to build its corresponding probabilistic model.
4. Property verification. Checks system properties against the obtained probabilistic models. To verify properties, we need to translate the probabilistic models obtained for each of the non-conventional operational profiles into an appropriate language to be used as input to a probabilistic model checker. Concretely, we use PRISM [3], which enables the verification of PCTL formulas against DTMC models.

4 Towards Run-Time Resilience Evaluation

The aim of the approach described in the previous section is enabling engineers to obtain an estimation of the system's resilience. However, the evaluation process presented above is sequential and external to the feedback control loop of the self-adaptive manager, preventing the incorporation of run-time information in the decision-making process regarding adaptation. Since our goal is to exploit the feedback provided by the evaluation process to improve the operation of the self-adaptive system (*e.g.*, safety, performance), we propose the integration of the resilience evaluation process into the feedback control loop of the self-adaptive manager. Concretely, our proposal consists in embedding a second type of loop running in parallel with the MAPE-K loop (Figure 2). The purpose of such loop (referred in the following as MAV - Monitor/Aggregate/Verify) is the continuous update of the quantified probabilities associated to system properties to be associated with the analysis and planning activities of the MAPE-K loop. However, this approach raises new challenges, namely:
Probabilistic model reliability. For evaluating resilience during development-time, a specifically tailored environmental changeload to trigger adaptation is used for stimulating the system's environment. In contrast, during run-time resilience evaluation, the evaluation has to rely on the conditions prescribed by the environment of the system. A major advantage for providing the means for collecting and analyzing information at run-time based on the actual operational conditions of the system, rather than

[3] http://www.prismmodelchecker.org

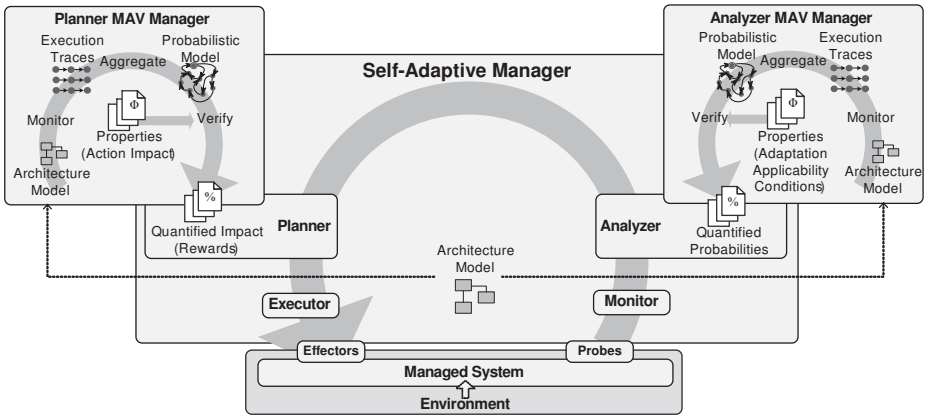


Fig. 2. Overview of run-time resilience evaluation.

changeload-based stimulation, is that in the long run high levels of confidence can be obtained since over time models will be adjusted to the drifting conditions of the system environment. However this has some drawbacks since the information about the system's behavior will be scarce during the initial period of the operational lifetime of he system and hence, the probabilistic models have to adjust before starting to yield reliable probability measures. A major challenge in this area is determining how reliable are probability estimations, based on the set system observations available (*e.g.*, in terms of coverage, etc.).

Integration. The main benefit of introducing a MAV loop for obtaining probabilities in the self-adaptive manager is that the MAPE-K loop can employ this information to make better informed decisions. However, making probabilistic information about properties available to the different MAPE-K activities is only the simpler aspect of integrating MAV into MAPE-K. The most challenging aspect consists in revisiting MAPE-K activities, approaching them in such a way that they can benefit from this new feedback: **(i)** analysis can benefit from sophisticated triggering conditions for adaptation, which may include probabilities (in contrast with simpler options, such as the violation of a constraint of the architecture model). For instance, an adaptation to reduce the number of operating servers in a system can include a condition such as *"The probability of exceeding the system's operating budget in the next hour is above 90%"*; and **(ii)** planning can benefit from estimations of the impact of carrying out a particular action on the system during adaptation, such as: *"The expected decrease on response time below* `MAX_RSPTIME` *in 60 seconds is X if server A is activated"*. In this example, the model checker can quantify [4] the value of X, and this kind of information can be incorporated when choosing the best actions for planning adaptation.

Deployment and Efficiency. The activities in the MAV loop have to be carried out in parallel with those in MAPE-K. However, the Analyzer and Planner need to evaluate different kinds of properties, so each one of them must have a dedicated MAV loop that is going to control the concrete MAPE-K activity by adjusting the probabilistic param-

[4] Impact is quantified by extending DTMC models with *rewards* [9].

eters used in it. This requires the use of a separate model-checking engine to carry out verification in each one of the MAV loops. In the case of model generation, a common aggregator module from which the MAV managers can extract probabilistic models can help to avoid duplicating model construction, even if aggregation is separated at the conceptual level in the two MAV loops (Figure 2). Moreover, one needs to consider that probabilistic model checking is a resource-demanding process, though there are some approaches that already tackle efficient probabilistic model checking at run-time [4]. However, they still have some limitations regarding their functionality with respect to full-fledged probabilistic model-checkers, such as PRISM. Anyway, continuous run-time evaluation can have a dramatic impact on performance, so efficient algorithms have to be encompassed with careful optimization of the models and smart scheduling of verification (*e.g.*, avoid recomputing probabilities if no significant changes in the corresponding probabilistic model have occurred).

5 Conclusions

Our immediate goal is tackling the aforedescribed challenges, validating our approach by endowing Rainbow, a platform for architecture-based self-adaption [5], with probabilistic evaluation capabilities (integrating PRISM as engine for verification of probabilistic properties). Other medium-term challenge is the provision of assurances regarding resilience in systems where goals change during operation, in contrast with our current approach, which assumes systems with fixed goals.

Acknowledgements. Co-financed by the Foundation for Science and Technology via project CMU-PT/ELE/0030/2009 and by FEDER via the "Programa Operacional Factores de Competitividade" of QREN with COMPETE reference: FCOMP-01-0124-FEDER-012983.

References

1. C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.
2. J. Cámara and R. de Lemos. Evaluation of Resilience in Self-Adaptive Systems Using Probabilistic Model-Checking. In *SEAMS*. IEEE, 2012. To appear.
3. B. H. C. Cheng et al. Software Engineering for Self-Adaptive Systems: a Research Roadmap. In *SEfSAS*, volume 5525 of *LNCS*, pages 1–26. Springer, 2009.
4. A. Filieri, C. Ghezzi, and G. Tamburrelli. Run-time efficient probabilistic model checking. In *ICSE '11*, pages 341–350, New York, NY, USA, 2011. ACM.
5. D. Garlan et al. Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure. *IEEE Computer*, 37(10):46–54, 2004.
6. L. Grunske. Specification Patterns for Probabilistic Quality Properties. In *ICSE*, pages 31–40. ACM, 2008.
7. S. Hart, M. Sharir, and A. Pnueli. Termination of Probabilistic Concurrent Programs. In *POPL*, pages 1–6. ACM, 1982.
8. J. Kephart and D. Chess. The vision of autonomic computing. *Computer*, 36:41–50, 2003.
9. M. Kwiatkowska, G. Norman, and D. Parker. Stochastic model checking. volume 4486 of *LNCS*, pages 220—270. Springer, 2007.
10. J.-C. Laprie. From Dependability to Resilience. In *DSN Fast Abstracts*. IEEE CS, 2008.
11. P. Oreizy et al. An architecture-based approach to self-adaptive software. *IEEE Intelligent Systems*, 14:54–62, May 1999.

Prototyping Component-Based Self-Adaptive Systems with Maude

Juan F. Inglés-Romero[1], Cristina Vicente-Chicote[1], Javier Troya[2], Antonio Vallecillo[2]

[1] Dpto. Tecnologías de la Información y Comunicaciones, E.T.S.I. de Telecomunicación, Universidad Politécnica de Cartagena, Edificio Antigones, 30202 Cartagena, Spain
{juanfran.ingles, cristina.vicente}@upct.es
[2] GISUM/Atenea Research Group. Universidad de Málaga, Spain
{javiertc, av}@lcc.uma.es

Abstract. Software adaptation is becoming increasingly important as more and more applications need to dynamically adapt their structure and behavior to cope with changing contexts, available resources and user requirements. Maude is a high-performance reflective language and system, supporting both equational and rewriting logic specification and programming for a wide range of applications. In this paper we describe our experience in using Maude for prototyping component-based self-adaptive systems so that they can be formally simulated and analyzed. In order to illustrate the benefits of using Maude in this context, a case study in the robotics domain is presented.

Keywords. Self-adaptation, component-based architecture, prototyping, Maude

1 Introduction

Nowadays, significant research efforts are focused on advancing the development of (self-) adaptive systems. In spite of that, some major issues remain still open in this field [1][2]. One of the main challenges is how to formally specify, design, verify, and implement applications that need to adapt themselves at runtime to cope with changing contexts, available resources and user requirements.

Adaptation in itself is nothing new, but it has been generally implemented in an ad-hoc way, that is, developers try to predict future execution conditions and embed the adaptation decisions needed to deal with them in their application code. This usually leads to increased complexity (business logic polluted with adaptation concerns) and poor reuse of adaptation mechanisms among applications [1]. The use of formal methods can help alleviating the limitations of current approaches to self-adaptive system development. In particular, they can provide developers with (1) a means for creating and sharing common foundations, based on their experience in self-adaptive system design; and (2) rigorous tools for testing and assuring the correctness of the adaptive behavior of their systems. The latter is a remarkable open issue, since only a few research efforts seem to be focused on the formal analysis and verification of self-adaptive systems.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 609-622, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

Maude [3] is a high-performance reflective language and system supporting both equational and rewriting logic specification and programming for a wide range of applications. The rewriting logic of Maude is simple, yet very expressive. This gives Maude good representational capabilities as a semantic framework to formally represent a wide range of systems, including distributed and concurrent systems, network protocols, etc. Maude and its supporting tools can be used in three, mutually reinforcing ways: as a declarative programming language, as an executable formal specification language, and as a formal verification framework.

A Maude program can be seen as an executable mathematical model of a system. Thus, using Maude for prototyping self-adaptive systems, enables their simulation, formally analysis (e.g., reachability/likelihood of certain system configurations) and verification (e.g., testing that the system reaches a consistent configuration for all given contexts). Furthermore, if the Maude prototype is simple, detailed and efficient enough, it could be directly used as the final system implementation.

In this paper we present our experience in using Maude for prototyping component-based self-adaptive systems. The results of this work derive from the implementation of a case study in the robotics domain. This research continues our previous works on self-adaptive system design [4] and implementation [5] using the framework developed within the EU 7FP DiVA Project [6].

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 details the proposed case study. Section 4 describes the Maude specification for modeling the self-adaptation logic of the case study. Section 5 reports the lessons learned and, finally, section 6 concludes and presents some future research lines.

2 Related Works

Significant research efforts are being invested to try overcoming the limitations of current ad-hoc approaches to (self-) adaptive system development. These efforts have given rise to new adaptation-enabling frameworks and middlewares, and new languages supporting adaptation primitives [2]. Some contributions provide a conceptual guidelines describing the different stages of self-adaptation [7], while others focus on the specification of design patterns for adaptive systems [8] [9]. Kramer et al. [10] propose a three-layer architecture for self-managed systems, and Oreizy et al. [11] present an infrastructure that simultaneously supports system adaptation and evolution. However, most current approaches do not offer either a formal specification of the adaptation processes, nor a formal reasoning support for testing, assessing and verifying the adaptation logic. This issue has been highlighted as a major challenge in several works [1][2].

The field of formal methods is very broad and there is a vast literature describing their many applications in different domains. The remaining of this section will focus on formal approaches targeting specific aspects of self-adaptation.

In the area of Architectural Description Languages (ADLs) supporting system adaptation, Wermelinger and Fiadeiro [12] present an algebra for formally specifying runtime architecture reconfiguration. Canal et al. [13] propose the use of LEDA (an

ADL supporting inheritance and dynamic reconfiguration) to specify dynamic programs. LEDA is based on the π-calculus, a simple but powerful process algebra that allows to automatically deriving prototypes from the specification. At a higher level of abstraction, Zhang et al. [14] present a model-based approach for formally specifying the behavior of adaptive programs, starting from high-level requirements. The resulting models can be analyzed using model checking techniques and can be used to generate rapid prototypes from them. Aligned to the latter, the work by Sama et al. [15] proposes a model-checking approach for detecting faults caused either by erroneous adaptation logic, or by the asynchronous updating of the context information that leads to inconsistencies between the physical context and its internal representation in the application. This proposal relies on the formalism provided by the Finite-State Machine theory. Cansado et al. [16] propose a formal framework that supports behavioral adaptation and structural reconfiguration. This approach relies on the formalisms provided by Labeled Transition Systems and model checking for (1) reasoning about whether it is possible or not to reconfigure the system; and (2) to verify certain reconfiguration-related properties. Weyns et al. [17] present a rigorous specification of a reference model for self-adaptation (called FORMS), defined using the Z notation. FORMS aims to (1) establish a shared vocabulary of primitives that can be used to precisely define arbitrary complex self-adaptive systems; (2) enable engineers to precisely express their design choices and assess them; (3) allow for comparison and evaluation of different types of self-adaptive systems; and (4) lay the foundation for a systematic method of developing a catalog of architectural patterns. Bruni et al. [18] propose the use of Maude to demonstrate the feasibility of their conceptual framework in which adaptation revolves around control data. The authors use the formal toolset provided by Maude (in particular, the statistical model checker PVesta) for simulation and analysis. As a case study, they consider an example based on robot swarms equipped with obstacle-avoidance and self-assembly capabilities.

3 Case Study

In this section, we introduce a robotic case study designed to illustrate the benefits of using Maude for prototyping self-adaptive systems. Firstly, we describe the adaptation scenario. Then, we briefly present the infrastructure we used to implement the case study. Finally, we provide some details about the component-based software architecture designed to cope with self-adaptation in the case study.

3.1 Adaptation Scenario

The case study takes place in a room (simulated with Lego blocks) where a small robot moves around randomly avoiding obstacles. In order to improve this basic functionality in terms of safety, power consumption and efficiency, the robot follows an adaptation strategy that decides on the following variation points: (1) the signaling type; (2) the signaling intensity; and (3) the robot velocity. There are two possible variants for the signaling type (namely, light or acoustic), while the signaling intensity

and the robot velocity may take any integer value in the range 0-100. The adaptation strategy decides the best possible configuration (selection of variants for each variation point) according to the current context. The context variables considered in the case study are the ambient light, the ambient noise and the robot battery level, all of them integer values ranging from 0 to 100.

The goodness of each configuration is calculated based on the impact of each variant on the three properties being considered, that is: safety, power consumption and efficiency. The following considerations are made concerning *safety* (making others aware of the presence of the robot in the surroundings): (1) light signaling is more convenient than acoustic signaling when the ambient light is low; and (2) the higher the ambient noise (might indicate a crowded environment), the higher must be the signaling intensity and the lower the robot velocity. Regarding *power consumption*, the greater the signaling intensity and the robot velocity the greater the power consumption. Thus, if the battery level is low, both the velocity and the signaling intensity need to be limited. Finally, concerning *efficiency*, the higher the velocity the shorter the time it takes to the robot to reach its goal position. Obviously, maximizing safety and efficiency, while simultaneously minimizing power consumption, imposes conflicting requirements. Thus, the adaptation strategy will need to find the right balance among these requirements to achieve the best possible configuration for a given context, even if some (or none) of them are optimized individually.

3.2 Case Study Implementation

As previously stated, we intend to use the versatility of rewrite theories and, in particular, of Maude for prototyping self-adaptive systems. However, the computational limitations of the experimental robot we selected as our target platform made it impossible for us to deploy the whole application in it. As a consequence we decided to adopt the remote processing schema illustrated in Figure 1.

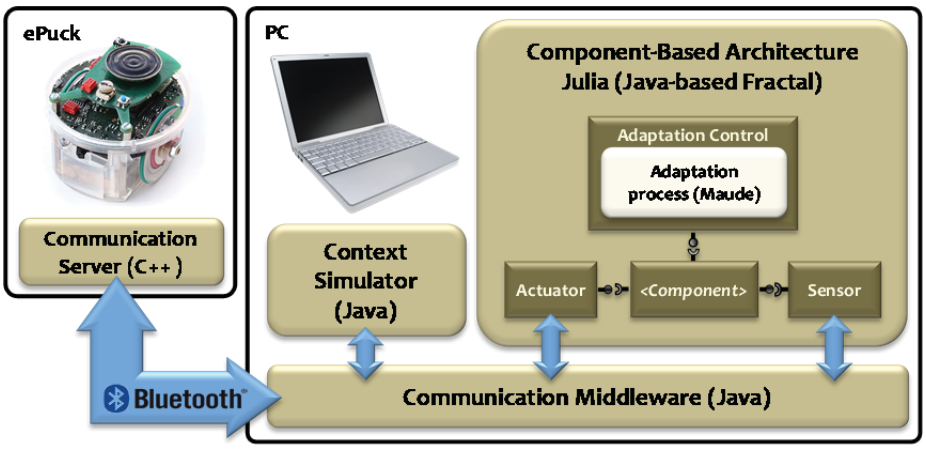


Fig. 1. Deployment infrastructure: core devices, applications and components

We selected the E-puck robot [19] as our target platform. E-pucks are low-cost mobile robots with a large range of sensors and actuators that make them appropriate for testing the proposed self-adaptation strategy. The E-puck robot runs a server that communicates with a PC via Bluetooth. This server provides the PC with information about the robot sensor status. Besides, it executes the low-level commands it receives from the PC (e.g., to turn on/off the lights or the speakers, to move faster or slower, etc.). In turn, the PC runs three applications (see Figure 1), namely: (1) the self-adaptive system architecture, specifically developed for the proposed case study; (2) a generic context simulator; and (3) a generic communication middleware. The self-adaptive architecture developed for the case study was implemented using Julia: the Java reference implementation of the Fractal component model [20]. We selected Julia for its runtime reconfiguration capabilities, among other interesting features.

In order to make the PC applications as independent as possible from the selected robotic platform and from each other, we have developed a generic (case study independent) communication middleware. This middleware provides the PC applications with a standard interface to interact with the robot services as if they were local. Besides, it manages process concurrency and offers flexibility to transparently connect different applications, e.g., a virtual robot (instead of a real one), an application displaying execution statistics, etc. Related to this, we have developed a generic (case study independent) context simulator to emulate changes in (some of) the context variables. In particular, in our case study, we simulate the robot battery level as E-pucks do not have a battery sensor. Even if this sensor was available, the simulation of this context variable seems more practical than waiting for the battery to drain, and having to recharge it before running the next adaptation test.

3.3 Component-Based Software Architecture

The component-based software architecture developed for the case study is sketched in Figure 2. As other self-adaptive systems [2], the proposed design includes: (1) a reconfigurable part, comprising the optional and/or parameterized components; (2) a set of monitoring components; and (3) an adaptation control unit.

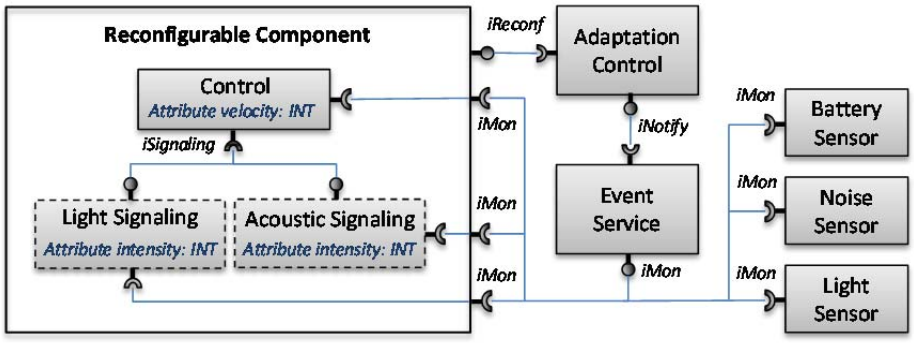


Fig. 2. Component-based software architecture for the case study

The *Reconfigurable Component* gathers the elements of the system that are susceptible to change at runtime. Among them, the *Control Component* implements the core robot functionality, that is, the motion control and the obstacle avoidance. This component includes a parameter called *velocity* that regulates the robot motion speed, and is responsible for activating or deactivating the robot signaling through the *iSignaling* interface. The *Reconfigurable Component* also contains two optional components, each one implementing one of the alternative ways for signaling the robot position: *Light Signaling* and *Acoustic Signaling*. Both these components contain an *intensity* parameter that regulates the frequency of the light and the acoustic signals, respectively. The three variation points available at the *Reconfigurable Component* (i.e., selecting one of the two alternative signaling components and setting the velocity and the intensity parameters) will need to be fixed at runtime by the adaptation strategy (implemented by the *Adaptation Control* as detailed later).

The monitoring part of the architecture provides the context-aware support for the adaptation. It comprises (1) a set of sensors (*Noise Sensor*, *Light Sensor* and *Battery Sensor*) and monitors (*Control*, *Light Signaling* and *Acoustic Signaling*) for acquiring information both from the environment (external context) and from the system itself (internal context); and (2) the *Event Service* component that receives the context information from the former components via the *iMon* interface, and notifies the changes in the context to the *Adaptation Control* component through the *iNotify* interface.

Finally, the *Adaptation Control* component implements the adaptation strategy which, on the basis of the context changes notified by the *Event Service* component, decides which is the best possible configuration (variant selection) for the *Reconfigurable Component* and applies the required changes via the *iReconf* interface. Next section details how the *Adaptation Control* component relies on Maude for executing this adaptation strategy.

4 Prototyping Self-Adaptation with Maude

This section describes the Maude specification of the self-adaptation strategy defined for the case study. Note that, for lack of space, we do not provide the complete Maude specification, but only the essential concepts for modeling the self-adaptation logic.

4.1 Overall Proposed Approach

Similarly to the systems described in [18], we have implemented our case study with Core Maude using an object-based programming approach. This allows us to model our self-adaptive systems as configurations (collections) of objects and messages that represent (a snapshot of) a possible system state. Each object has an identifier, a class and a set of attributes (e.g., `< oid : cid | attr1, attr2 >` represents an object with identifier `oid`, belonging to the class `cid`, and with two attributes `attr1` and `attr2`). On the other hand, messages are described as operators that return a value of type `Msg`. Each message includes an identifier and a list of arguments (e.g., `mid(arg0, arg1)` represents a message with identifier `mid` and arguments `arg0` and

`arg1`). The idea behind using a set of objects and messages to represent the system state is that we can specify the adaptation behavior as a set of rewrite rules that consume and produce objects and messages, i.e., evolve the system state.

We have used Maude for specifying both the main adaptation loop and a bridge aimed to enable the communication between Maude and the Java implementation of the *Adaptation Control* component. Regarding the later, the communication is performed through the standard I/O using the Domain Specific Language (DSL) summarized in Table 1. A *read-eval-print* loop has been implemented to handle this communication and to maintain the persistent state of the application.

Concerning the adaptation loop, as in most self-adaptive systems [2], it mainly comprises three processes, namely: (1) gathering and assessing the current context, (2) reasoning on the best adaptation possible, and (3) performing the system reconfiguration. In our case, all these processes are carried out by the *Adaptation Control* component. Figure 3 outlines the steps of the algorithm that implements this adaptation loop. Each of these steps is further detailed in the following subsections.

Table 1. DSL for the interaction between Maude and the *Adaptation Control* component

Command	Description
Start	Starts the adaptation loop
synchArch <*component : String*> <*parameter : String*> <*value : String*>	Synchronizes the Maude architecture representation with the actual Fractal architecture implementation. Example: *synchArch "LightSignaling" "state" "running"* → Maude is notified of the actual state of the *LightSignaling* comp.
init (<*battery : INT*>, <*noise : INT*>, <*light : INT*> *)*	Maude is notified of the initial low-level context variables Example: *init (100, 55, 20)*
battery <*value : INT*>	Updates the battery (0-100). Example: *battery 23*
noise <*value : INT*>	Updates the ambient noise (0-100) Example: *noise 67*
light <*value : INT*>	Updates the ambient light (0-100) Example: *light 10*
notify <*component : String*> <*parameter : String*> <*value : String*>	Maude is notified of a change in a component. Example: *notify "control" "velocity" "23"* → The control component notifies that the velocity has changed to 23
command <*component : String*> <*parameter : String*> <*value : String*>	Maude sends a reconfiguration command. Example: *command "control" "velocity" "11"* → The velocity of control component must be changed to 11

4.2 Initialization

Prior to starting the adaptation loop, an initialization function needs to set up the context and the architecture models that will be used throughout the adaptation process. This function is labeled in Figure 3 as "*Init context and synchronize representation*".

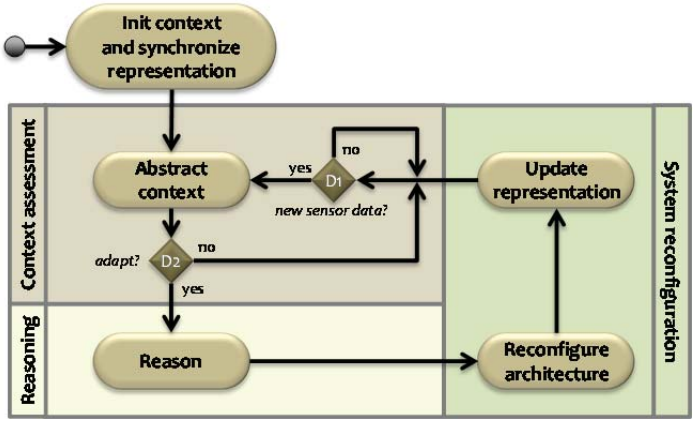


Fig. 3. Outline of the adaptation loop

The context model. To safely and efficiently adapt a running system, it is important to have a well-fitted model of its context. A context model should be both detailed enough to gather all the contextual information relevant for the adaptation, and abstract enough to enable the system to efficiently reason on it. The proposed case study considers three low-level context variables, namely: the robot battery level and the ambient noise and light levels. The context model abstracts these low level variables by defining three new high-level variables: *batt* ∈ {*LOW*, *MEDIUM*, *FULL*}; *noise* ∈ {*NORMAL*, *NOISY*, *NOISIEST*}; and *light*, which is a Boolean. The function that computes the high-level context variables from the low-level ones (e.g., deciding when a *battery level* is LOW, MEDIUM or FULL) will be later detailed in section 4.3. A possible configuration of the context model could be as follows:

```
< ctx : Context | batt  : FULL, noise : NORMAL, light : false >
```

The architecture model. Similarly to the context model, maintaining an explicit reflection model that abstracts the actual running system is essential to efficiently decide on and execute the required reconfigurations. This model needs to be synchronized with the actual component-based system architecture in order to provide the adaptation logic with up-to-date information. With regard to adaptation, the only relevant information contained in the case study system architecture (see Figure 1) is the list of components gathered in the *Reconfigurable Component* (neither the component interfaces nor the connectors are modeled). Each component in this list is modeled in Maude with an object containing, at least, two attributes: *name* (String) and *state* ∈ {*RUNNING*, *STOPPED*}. An additional attribute will be added for each parameter defined in each component. A possible configuration of the architecture model could be as follows (please, note that the state of the AcousticSignaling component is STOPPED, meaning that it is not present in the actual system architecture):

```
<c : Control | name : "control", state : RUNNING, velocity : 5 >
<l : LightSignaling | name : "lsig", state : RUNNING, intensity : 50 >
<a : AcousticSignaling | name : "asig", state :STOPPED, intensity:50>
```

Architecture and context model initialization. Before starting the adaptation loop, the context and the architecture models need to be created and synchronized for the first time to reflect the actual situation. Firstly, Maude creates and initializes a default architecture model containing all the components in the *Reconfigurable Component*. Secondly, when the Fractal architecture is created, all its components send (via *iMon*) their initial state and attribute values to *Event Service*. This component notifies (via *iNotify*) these values to *Adaptation Control* which, in turn, sends to Maude one or more `synchArch` message for each component. These messages trigger in Maude the `arch-synchronization` rewrite rule, which consumes the message and updates the state and attributes of the corresponding components in the architecture model. Messages from components not belonging to the *Reconfigurable Component* are discarded and produce no update. Finally, Maude waits until the *Adaptation Control* sends an `init` message with the initial context information. This message triggers the `init-context` rewrite rule, which (1) consumes the message; (2) creates and initializes the context model; and (3) creates a `reasoner` message that launches the reasoning process (later discussed) to assure that the system adapts (if necessary) to the initial conditions. This starts the adaptation loop and, from that moment on, no more `init` or `synchArch` messages are accepted (if they arrive, they are automatically discarded).

4.3 Context Assessment

The main functions of the *Context Assessment* process are: (1) to update the low-level context variables when Maude receives new sensor data (see the `battery`, `noise` and `light` commands in Table 1); (2) to update the high-level context model from the low-level values previously received; and (3) to launch the reasoning process in case the changes in the high-level context variables are significant enough. These three functions are represented in Figure 3 in the decision node *D1*, the operation "*Abstract context*" and the decision node *D2*, respectively.

In order to compute the high-level context model from the low-level sensor data we have implemented three rewrite rules (one for each context variable). These three rules share the same structure: the left-hand side term contains the current context object and the message with the new low-level context value (this message is consumed once the rule is executed) and the right-hand side term contains the abstraction and the activation functions detailed next.

The abstraction function. This function maps the low-level context (variables usually quantified as integer or float values) into the high-level context (variables usually modeled as enumerations or Booleans, providing a more qualitative than quantitative information). In the current implementation, we use fix thresholds (predefined at design-time) for segmenting the low-level context data. For instance, we consider the *batt* (high-level) to be *FULL* when the *battery* (low-level) value is greater than 80.

The activation function. This function determines how much the context must change to require a new adaptation step. If this function is not appropriately adjusted at design-time it may lead to a slow or ineffective adaptation or, what is worse, to an

instable situation in which continuous reconfigurations are made to cope with every single small change in the context. In the current implementation, the adaptation process starts only if a high-level context variable changes. In this case, a `reasoner` message is created that triggers the rules performing the reasoning process, detailed next. This mechanism is more robust and stable than defining a fix variation range on a low-level context variable that, when exceeded, causes a new adaptation step.

4.4 Reasoning

The *Reasoning* function (see Figure 3) implements the core self-adaptation logic as it computes the best configuration possible for a given context, that is, it selects the set of abstract variants that jointly optimize the overall system performance (in our case the overall system safety, efficiency or power consumption). The abstract variants considered by Maude conform to the variability model described next.

The variability model. As detailed in section 3.1, the proposed case study considers three low-level variation points, namely: the signaling type (implies selecting the Light Signaling or the Acoustic Signaling component), the signaling intensity (integer ranging 0-100), and the robot velocity (integer ranging 0-100). The variability model abstracts these low-level variation points by defining three new high-level ones, namely: *signaling* ∈ {*LIGHT*, *ACOUSTIC*}; *intensity* ∈ {*LOW, MEDIUM, HIGH*}; and *velocity* ∈ {*SLOW*, *MEDIUM*, *FAST*}. The abstraction provided by this model, together with the one provided by the context and architecture models, significantly simplifies the reasoning process. As shown below, each abstract variant is modeled in Maude with an object containing the following attributes: *name*, *dimension* (ID of the high-level variation point the variant belongs to), *safety*, *consumption* and *efficiency* (impact of the variant in each property), *score* and *state*.

```
< v : Variant | name : "slow", dimension : "velocity", safety : 3,
  consumption : 2, efficiency : 1, score : 0, state : AVAILABLE >
```

The reasoning approach followed in this research is based on the method described in [21], which combines (1) the use of adaptation rules and (2) the optimization of property-based adaptation goals. Our adaptation rules have been implemented as two Maude rewrite rules, `non-available` and `required`. Both these rules are executed once for each variant object, updating its *state* attribute according to the current context model. The `non-available` rule sets the *state* of those variants that are inconsistent with the current context model (i.e., cannot be selected during the subsequent optimization process) as NON-AVAILABLE. For example, if the high-level *batt* context variable is not *FULL*, then the high-level variant *FAST* is marked as NON-AVAILABLE for the *velocity* variation point. The `required` rule sets the *state* of those variants that, according to the current context, need to be compulsorily selected as REQUIRED. For example, if the high-level *light* context variable is *true*, then the high-level variant *ACOUSTIC* is marked as REQUIRED.

In order to cope with the optimization of property-based adaptation goals, we have implemented two additional rewrite rules: `calculate-scores` and `search-`

`solution`. The first of these rules is triggered once for each variant and calculates the attribute *score* of those marked as AVAILABLE. The higher the *score* of a variant the better it fits the current context, i.e., the more likely to be selected as part of the new system configuration. The calculation of the *score* is based on: (1) the impact of each variant on the three system properties (ranging from 0: no impact to 5: very high impact); and (2) the importance of each property in the current context (also ranging from 0: no importance to 5: very high importance). The impact of each variant on the three properties is defined at design-time and stored in the abstract variant objects. The importance of each property depending on the context is also defined at design-time but, in this case, using Maude equations (e.g., there is an equation stating that if the *batt* is *LOW*, the importance of the *power consumption* property must be set to 5).

Finally, the `search-solution` rule finds the best possible system configuration for the current context, that is, the set of abstract variants that, together, obtain the highest score. This rule updates the *SystemConfig* object, which contains one attribute per high-level variation point. The following example shows the *SystemConfig* object resulting of a reasoning step in which the variants *LIGHT*, *MEDIUM* and *FAST* were respectively selected for the *signaling*, *intensity* and *velocity* variation points.

```
< c : SystemConfig | signaling : "light", intensity : "medium",
  velocity : "fast" >
```

4.5 System Reconfiguration

The main functions of the *System Reconfiguration* process are: (1) to create a reconfiguration plan (sequence of reconfiguration commands) that adapts the architecture model according to the decision made by the *Reasoning* function; and (2) to synchronize the architecture model with the runtime system architecture. To implement these functions, labeled in Figure 3 as "*Reconfigure architecture*" and "*Update representation*", we have implemented two Maude rewrite rules: `reconfigure` and `notification-when-pending`.

The `reconfigure` rule takes the *SystemConfig* object (updated by the *Reasoning* function) and the current architecture model (set of component objects) as its input, and produces a set of reconfiguration commands. In order to map the high-level variants, selected in the *SystemConfig* object, into the low-level ones, defined for the elements gathered in the *Reconfigurable Component*, we have defined a set of Maude equations (similar to those defining the relations among the system properties and the context variables). For instance, there is an equation that maps the *velocity* variant *FAST* with the value 90 for the attribute *velocity* of the *Control* component. When all the variants have been mapped, the rule generates the reconfiguration commands only for those components that need to be modified (i.e., those for which the state or other attribute has changed). This is achieved by making the difference between the current architecture model and the one that has just been derived from the selected variants.

The `notification-when-pending` rule is executed whenever a real component (belonging to the *Reconfigurable Component*) notifies that it has changed in response to a reconfiguration command. These notifications cause the architecture model to be

updated to reflect the current situation. It is worth noting that we use an *Adaptation-System* object to register all the reconfiguration commands sent by Maude and not acknowledged yet with the corresponding notification. Context messages are discarded while this object is not empty. This prevents the execution of new adaptation loops while the architecture model and the running system are not completely synchronized.

5 Lessons Learned

The first benefit of using Maude for prototyping self-adaptive systems stems from its capability to provide designers with executable mathematical models of these systems. This capability becomes essential for adjusting and validating their adaptation behavior. Specifically, Maude can assist designers in (1) adjusting the activation function (see section 4.3) to make the adaptation stable, avoiding continuous system reconfigurations; (2) adjusting the design-time values that define the impact of the variants on the system properties, and the weight of these properties depending on the context (see section 4.4); and (3) establishing the right links between the high-level and the low-level context variables (see the abstraction function in section 4.3) and between the high-level and the low-level variants (see section 4.5).

Regarding simulation, Maude enables the execution of the system specification starting from any given state. This can be very useful for addressing the adjustments enumerated above. For instance, to test the Activation Function, we could measure the system reactivity by recording the number of reconfigurations performed per time unit, and relating this number with the context variation rate. However, this kind of analysis usually requires including some additional terms in the specification. This not only pollutes the prototype with analysis-specific code but also may influence the analysis results, as it might affect the overall performance of the prototype.

Concerning model checking, Maude provides the `search` command, which explores the reachable state space looking for a given configuration. This command is a simple, yet very useful method for checking invariants. For example, consider the following statement: the state of the LightSignaling and AcousticSignaling components cannot be simultaneously RUNNING. If we use the `search` command to find a counterexample and it returns an empty answer then we can assure that the statement is never violated. Maude also provides other tools supporting more complex model checking capabilities, e.g., a module for linear time temporal logic. It is worth noting that, in general, model checking processes are highly memory- and time-consuming.

Regarding the reusability of the Maude specification, it is worth noting that, although it is application-dependent, there some common structures (described in section 4) that could be easily reused. Also related to reusability, it is worth highlighting that the proposed design (see Figure 2) follows the separation of concerns principle, since the adaptation logic (implemented in Maude and embedded in the *Adaptation Control* component) is explicitly separated from "business" logic (in our case study implemented in the components gathered in the *Reconfigurable Component*). The benefits of such a decision are twofold: on the one hand, it reduces the complexity

and improves the maintainability of the design and, on the other hand, it promotes the reuse and sharing of adaptation mechanisms among applications.

Some additional benefits of using Maude for prototyping self-adaptive systems are [3]: (1) the rapid development process and the reduced length of the programs, compared to other (traditional) programming languages. Prototyping in Maude has allowed us to focus on the development of the adaptation mechanisms without having to spend much time in implementation details; (2) the simplicity and versatility of using (a) a set of objects and messages to describe the system state; (b) a set of equations to model the system data; and (c) a set of concurrent rules to describe the system behavior; and (3) the performance of Maude prototypes is reasonably good. In fact, the performance of the prototype developed for our case study seems to be good enough for using it as part of a real self-adaptive system. The main limitation we found relates with the difficulty for debugging Maude programs, due to the concurrent nature of its rules and the scarcely legible traces it returns during the execution.

6 Conclusions and Future Work

This paper reports our experience in using Maude for prototyping, simulating and verifying component-based self-adaptive systems. In order to demonstrate the benefits (and also to assess the limitations) that Maude can bring in this field, we have developed a case study in the robotics domain that relies on a distributed processing schema. The robot adaptation logic, implemented in Maude, has been embedded (separate from the business logic) in one of the components of the Fractal-based implementation of the system. This component is fed with contextual information and controls the adaptation of the reconfigurable part of the architecture. In order to make the adaptation decisions efficient, the Maude specification works with abstract models of the context, the architecture and its variability. For the future, we plan to continue exploring the potentials of Maude, in particular, for verifying the completeness and correctness of the self-adaptive behavior specifications. We also plan to link this work with our previous experience with the model-driven approach proposed by DiVA [6]. In this sense, our intention is to generate the Maude specification from the DiVA models describing the self-adaptive system design.

Acknowledgements

This work has been partially funded by the EXPLORE (MICINN, TIN2009-08572) and the MISSION (Fundación Séneca-CARM, 15374/PI/10) projects. Juan F. Inglés-Romero thanks Fundación Séneca-CARM for a research grant (Exp. 15561/FPI/10).

References

1. Cheng, B.H.C., et al.: Software engineering for self-adaptive systems: A research roadmap. In: Cheng, B.H.C., et al. (Eds.), Software Engineering for Self-Adaptive Systems. LNCS, vol. 5525, pp. 1–26. Springer, Heidelberg (2009)

2. Salehie, M., Tahvildari, L.: Self-adaptive software: Landscape and research challenges. ACM Transactions on Autonomous and Adaptive Systems, 4(2), 1–42 (2009)
3. Clavel, M., et al.: All About Maude. A High-Performance Logical Framework: how to specify, program and verify systems in rewriting logic. Springer-Verlag (2007). ISBN 978-3-540-71940-3
4. Inglés-Romero, J. F., et al.: Using Models@Runtime for Designing Adaptive Robotics Software: an Experience Report. In: 1st Int'l Workshop on Model-Based Engineering for Robotics (RoSym), 3-8 October, Oslo, Norway (2010)
5. Inglés-Romero, J. F., et al.: Towards the Automatic Generation of Self-Adaptive Robotics Software: An Experience Report. In: 20th IEEE Int'l Conf. on Collaboration Technologies and Infrastructures (WETICE'), pp. 79–86, 27-29 June, Paris, France (2011)
6. EU 7FP DiVA Project, www.ict-diva.eu/
7. Kephart, J., et al.: The Vision of Autonomic Computing. Computer, 36(1), 41–50 (2003)
8. Gomaa, H., Hussein, M.: Software Reconfiguration Patterns for Dynamic Evolution of Software Architectures. In: 4th Working IEEE/IFIP Conference on Software Architecture (WICSA), pp. 79–88, 12-15 June, Oslo, Norway (2004)
9. Ramirez, A.J., Cheng, B.H.C.: Design Patterns for Developing Dynamically Adaptive Systems. In: 2010 Int'l Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), pp. 49–58, 3-4 May, Cape Town, South Africa (2010)
10. Kramer, J., Magee, J.: A Rigorous Architectural Approach to Adaptive Software Engineering. Journal of Comp. Science and Technology, 24(2), 183–188 (2009)
11. Oreizy, P., et al.: An architecture-based approach to self-adaptive software. IEEE Intelligent Systems, 14(3), 54–62 (1999)
12. Wermelinger, M., Fiadeiro. J.L.: Algebraic software architecture reconfiguration. In: Nierstrasz, O., Lemoine, M. (Eds.), Software Engineering Conference (ESEC/FSE'99). LNCS, vol. 1687, pp. 393–409, Springer, Heidelberg (1999)
13. Canal, C., Pimentel, E., Troya, J.M.: Compatibility and inheritance in software architectures. In: Science of Computer Programming, 41(2), 105-138 (2001)
14. Zhang, J., Cheng, B.H.C.: Model-based development of dynamically adaptive software. In: 28th Int'l Conf. on Software Engineering (ICSE), pp. 371–380, 20-28 May, China (2006)
15. Sama, M., Elbaum, S., Raimondi, F., Rosenblum, D.S., Wang, Z.: Context-Aware Adaptive Applications: Fault Patterns and Their Automated Identification. In: IEEE Trans. on Software Engineering, 36(5), 644–661 (2010)
16. Cansado, A., Canal, C., Salaün, G., Cubo, J.: A Formal Framework for Structural Reconfiguration of Components under Behavioural Adaptation. In: Electron. Notes Theor. Comput. Sci., 263, 95-110 (2010)
17. Weyns, D., Malek, S., Andersson, J.: FORMS: a formal reference model for self-adaptation. In: 7th International Conference on Autonomic Computing (ICAC), pp. 205–214, 7-11 June, Washington DC, USA (2010)
18. Bruni, R., et al.: Modelling and analyzing adaptive self-assembling strategies with Maude. In: 9th Int'l Workshop on Rewriting Logic and its Applications (WRLA), held in the context of the 2012 European Joint Conferences on Theory & Practice of Software (ETAPS), pp. 48-67, 24-25 March, Tallinn, Estonia (2012)
19. The E-puck website, http://www.e-puck.org
20. The Fractal Project, http://fractal.ow2.org/
21. Fleurey, F., Solberg, A. A Domain Specific Modeling Language Supporting Specification, Simulation and Execution of Dynamic Adaptive Systems. In: *12th Int'l Conf. on Model Driven Engineering Languages and Systems* (MODELS), pp. 606-621, 4-9 October, Denver, Colorado, USA (2009)

MinFrEditor: Entorno de desarrollo de aplicaciones para un framework de componentes*

Francisco Sánchez-Ledesma, Juan Pastor y Diego Alonso

División de Sistemas e Ingeniería Electrónica (DSIE), Universidad Politécnica de Cartagena, Campus Muralla del Mar, E-30202, Spain
francisco.sanchez@upct.es

Resumen: El entorno de modelado MinFrEditor integra un conjunto de herramientas diseadas para facilitar el desarrollo de aplicaciones basadas en componentes utilizando el framework MinFr.
Este entorno de desarrollo permite a los usuarios de MinFr: (1) Modelar aplicaciones basadas en componentes utilizando un leguaje textual, (2) hacer el despliegue de las aplicaciones y (3) generar modelos de entrada para herramientas de análisis de tiempo real.

Palabras clave: ingeniería de software; desarrollo de software basado en componentes; desarrollo de software dirigido por modelos; framework; editor; herramienta de desarrollo

1 Introducción

La mejor forma de abordar la complejidad en el desarrollo del software es elevar el nivel de abstracción de los elementos de modelado, construyendo los sistemas a partir de módulos independientes que interaccionan entre sí únicamente a través de sus interfaces. No existe una definición única de lo que es un módulo y de hecho cada paradigma de programación ofrece diferentes respuestas considerando diferentes niveles de abstracción y de granularidad. El enfoque elegido en este trabajo es el Desarrollo de Software Basado en Componentes (CBSD en sus siglas inglesas) , en el cual un componente software es una unidad de composición con interfaces bien definidas y un contexto de uso explícito.

En el marco anteriormente mencionado de desarrollo de software basado en componentes, se decidió seguir un enfoque de Desarrollo Software Dirigido por Modelos (MDSD en sus siglas inglesas) para modelar este tipo de aplicaciones para sistemas de tiempo real estricto, ya que MDSD es una tecnología que proporciona el soporte conceptual y tecnológico tanto para el modelado de las aplicaciones como para la generación final de código.

Aunque el presente artículo se centra en la en la descripción de la herramienta de modelado es conveniente entender el contexto para comprender la utilidad de

* Este proyecto ha sido financiado parcialmente por el proyecto CICYT EXPLORE (ref. TIN2009-08572), El proyecto Séneca MISSION-SICUVA (ref. 15374/PI/10), y la beca MEC FPU(AP2009-5083).

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 623-626, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

la herramienta. En un trabajo anterior se definió el lenguaje de componentes V3CMM [1] como un meta modelo que proporciona al diseador tres vistas complementarias y débilmente acopladas: (1) una vista arquitectónica para definir componentes (interfaces, puertos, servicios ofrecidos y requeridos, componentes compuestos, etc.) , (2) una vista para especificar el comportamiento de los mismos utilizando autómatas temporizados, y finalmente (3) una vista algorítmica para expresar la secuencia de acciones que ejecuta un componente en función de su estado.

En un trabajo posterior [2] se desarrolló e implementó un framework OO, denominado MinFr, que proporciona las clases base para implementar los componentes del diseo arquitectónico definidos con V3CMM, y una infraestructura para que el usuario elija las características de concurrencia que finalmente quiere para su aplicación: número de tareas, código que ejecuta cada una de ellas, plazos, prioridades, periodos, etc. MinFr proporciona una interpretación OO de los conceptos CBSD que permite trasladar los diseos basados en componentes a aplicaciones OO utilizando para ello sus hot-spots.

Mediante el uso de MDSD se han desarrollado, en la plataforma de libre distribución Eclipse, el MinFrEditor que es un plug-in que facilita el desarrollo de aplicaciones que utilizan el framework antes mencionado. Este entorno integra un conjunto de herramientas que permiten:(1) modelar aplicaciones que utilizan directamente el MinFr utilizando un leguaje textual, (2)generar modelos de análisis de tiempo real y (3)hacer el despliegue de las aplicaciones en uno o más nodos computacionales.

2 Descripción de MinFrEditor

A continuación se describen las tres herramientas que conforman MinFrEditor, las cuales fueron desarrolladas utilizando las facilidades que ofrece la plataforma eclipse y el lenguaje de programación java. Primero, en la la Sección 2.1, se describe la herramienta de modelado. A continuación, la Sección 2.2, se explica la transformación modelo a texto utilizada para generar los modelos de análisis de tiempo real. Por último, la Sección 2.3, se explica como se hace la distribución de la aplicación en uno o varios nodos computacionales.

2.1 Herramienta de modelado.

Para dar soporte al MinFr se han creado un meta modelo (ver figura 1) haciendo uso de las facilidades que nos proporciona el EMF. En este meta modelo se incluyen todos los conceptos necesarios para modelar las aplicaciones que utilizan el MinFr. Tomando como base el meta modelo de MinFr se creó un leguaje textual utilizando la herramienta xtext .

El MinFrEditor proporciona un editor para el leguaje textual (ver figura 2), con coloración de sintaxis y auto-compleción de código, que permite crear aplicaciones de tiempo real basadas en componentes. En concreto el editor permite crear tres tipos de modelos diferentes que son:(1) Librerías de componentes, en

este modelo se definen componentes los que conformarán las aplicaciones. Las Librerías sirven como base para el resto de los ficheros. (2)La Aplicación, en este modelo se hacen la instaciación de los componentes que han sido definidos previamente en las librerías y la conexión de sus puertos. Y (3) el despliegue, en este modelo se define como se va a desplegar la aplicación. Se asignan las instancias de componentes a nodos y procesos, y la regiones a los hilos.

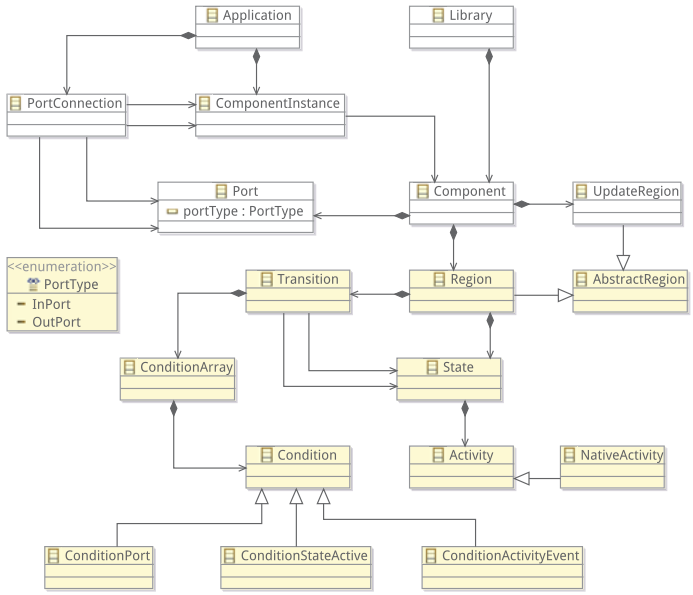


Fig. 1. Meta-Modelo simplificado del framework MinFr

2.2 Generación del modelo de análisis de tiempo real

A partir de los modelos creados utilizando los editores del lenguaje textual se puede generar un modelo de análisis de tiempo real que sirve como entrada en la herramienta de análisis Cheddar [3]. Dado el formato XML que sigue la herramienta, en este caso se ha desarrollado una transformación modelo a texto utilizando el lenguaje Xtend para generar el fichero con la información que necesita Cheddar para llevar a cabo el análisis. En el modelo de despliegue se definen de forma explícita los hilos (Threads), que corresponden directamente con las actividades del modelo Cheddar (Tasks) . La transformación Xtend estima el periodo de cada uno de estos hilos como el máximo común divisor de todos los periodos de todas las regiones asignadas al hilo. Cada hilo se comporta en MinFr como un ejecutivo cíclico en miniatura y, por tanto, es posible asignar regiones con distintos períodos al mismo hilo. .

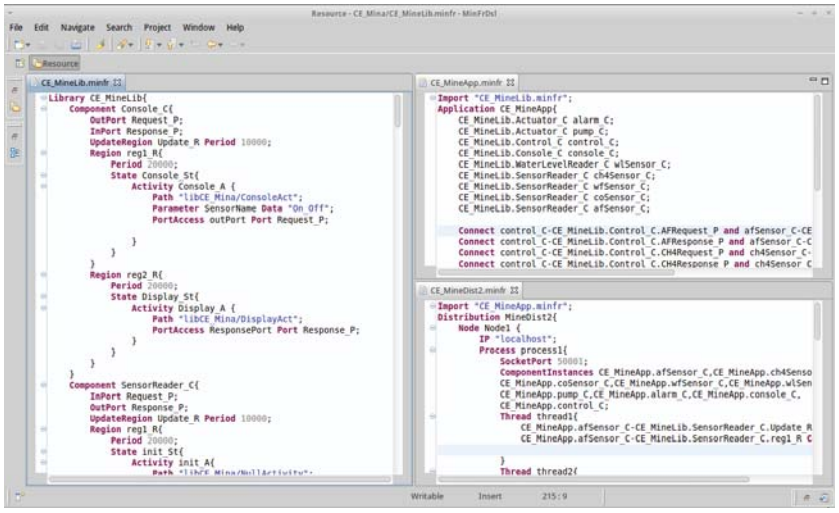


Fig. 2. Entorno de desarrollo MinFrEditor

2.3 Despliegue de la aplicación

Para facilitar el despliegue de las aplicaciones desarrolladas con MinFr en nodos y procesos se ha creado un programa utilizando el leguaje de programación java y se ha intregrado al MinFrEditor. Este programa carga el modelo de despliegue, se conecta por socket con los nodos indicados en el modelo, le envía a los nodos el modelo de los componentes deben instanciar cada nodo y las indicaciones necesarias para que los nodos se conecten entre si utilizando socket y conecten los puertos de los componentes que correspondan. Para que este proceso sea posible cada nodo de la red debe estar ejecutando una instancia de MinFr que es la que va a recibir las instrucciones y también deben tener instaladas las librerías de actividades que utilicen.

Referencias

1. Alonso, D., Vicente-Chicote, C., Barais, O.: V3Studio: a Component-Based architecture modeling language. In: 15th Annual IEEE International Conference and Workshop on Engineering of Computer Based Systems, ieee (March 2008) 346–355
2. Pastor, J., Alonso, D., Sanchez, P., Alvarez, B.: Towards the definition of a pattern sequence for Real-Time applications using a Model-Driven engineering approach. In: Proc. of the 15th Ada-Europe International Conference on Reliable Software Technologies, Ada Europe 2010. lncs, spr (June 2010) 167–180
3. Singhoff, F., Plantec, A., Dissaux, P., Legrand, J.: Investigating the usability of real-time scheduling theory with the cheddar project. Journal of Real Time Systems **43**(3) (November 2009) 259–295

Agile Product Line Engineering—A Systematic Literature Review[1]

Jessica Díaz, Jennifer Pérez, Pedro P. Alarcón, Juan Garbajosa
Technical University of Madrid (UPM) - Universidad Politécnica de Madrid
E.U. Informática Ctra. Valencia Km. 7 E-28031 Madrid, Spain
yesica.diaz@upm.es,{jenifer.perez,pedrop.alarcon,jgs}@eui.upm.es

1. Summary

Software Product Line Engineering (SPLE)[1] demands upfront long-term investment in (i) designing a common set of core-assets and (ii) managing variability across the products from the same family. When anticipated changes in these core-assets have been predicted with certain accuracy, SPLE has proved significant improvements. However, when large/complex product-line projects have to deal with changing market conditions, alternatives to supplement SPLE are required.

Agile Software Development (ASD)[2] may be an alternative, as agile processes harness change for the customer's competitive advantage. However, when the aim is to scale agile projects up to effectively manage reusability and variability across the products from the same family, alternatives to supplement agility are also required. As a result, a new approach called agile product line engineering (APLE) [3] advocates integrating SPLE and ASD with the aim of addressing these gaps.

APLE is an emerging approach, what implies that organizations have to face with several barriers to achieve its adoption. This paper[1] presents a systematic literature review of experiences and practices on APLE, in which the key findings uncover important challenges about how to integrate the SPLE model with an agile iterative approach to fully put APLE into practice. From this systematic literature review two main ideas can be highlighted. First, practitioners can conclude that there are sufficient reasons to move towards a combination of SPLE and ASD. And second, researchers can conclude that there are still some important challenges in the area, and therefore, more research work is required to completely put into practice APLE.

With regard to the first conclusion, the literature review reports that APLE would be applicable to business situations in which the convenience of going towards a product line has been identified, but at the same time the market situation is not enough stable for different reasons, including technological factors. Specifically, the review has identified four main advantages of putting APLE into practice, and when it is advantageous: (1) If SPL developers do not have enough knowledge to completely perform the DE[2], ASD may facilitate the elicitation of further knowledge. (2) Trade-offs between SPLE and ASD provide the opportunity to apply the APLE approach to

[1] Jessica Díaz, Jennifer Pérez, Pedro Pablo Alarcón, Juan Garbajosa: *Agile product line engineering - a systematic literature review*. Journal Software Practice and Experience. 41(8): 921-941, July 2011. Wiley InterScience, 2011 [DOI:10.1002/spe.1087 JCR 0.71]

[2] Domain Engineering process [1]

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 627-628, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

a wider variety of projects than those served by only applying ASD or SPL methods. (3) When anticipated changes cannot be predicted and the product life-cycle is not known, it would be advantageous to use an incremental approach such as APLE. (4) Agile processes may facilitate fast feedback cycles between requirements engineering, development, and field trial in innovative business.

With regard to the second conclusion, the literature review has reported four main findings:

- The applicability of agile methods to DE requires more effort than the application of AE[3]. This is due to the fact that is difficult to reduce the upfront design with the aim of getting closer to agile principles and values, while achieving the typical goals of DE such as reuse. Nevertheless, the applicability of agile methods to AE does seem feasible.
- Synchronization between platform and product teams (DE and AE teams, respectively) is vital in APLE, as DE and AE should not be separated. The platform should be synchronized with the application needs to avoid that the platform will become obsolete. It is still a challenge for APLE engineers.
- Business objectives should be used to identify the right level of flexibility. This level should be useful to determine the combination of SPLE and ASD: either SPLE and ASD at strategic level, or SPLE at strategic level and ASD at tactical.

Since this systematic literature review pursues to be useful for both practitioners and researchers, the analysis of the most relevant findings have been classified into two points of view: *How can APLE be put into practice?*, i.e. what problems have been solved in APLE deployment and which approaches/practices can be considered for APLE adoption; and *Which are the open research challenges?*, i.e. what is still missing and what challenges/barriers researchers should cope with? Regarding the open challenges, this review reports that the potential challenges and risks associated with APLE are the following: (1) traceability management and maintenance of components might be difficult in agile approaches without explicit knowledge; and (2) if PL Architectures are tailored to be more Agile, there is a danger that a valuable architecture supporting other products of the family may be damaged. As a result, APLE architecture and APLE traceability are still open research challenges.

Acknowledgment. The work reported here has been partially sponsored by the Spanish MEC (DSDM TIN2008-00889-E), MICINN (INNOSEP TIN2009-13849), and by UPM (Researcher Training program).

References

1. Pohl K, Böckle G, Linden F. Software Product Line Engineering: Foundations, Principles and Techniques. Springer: Germany, 2005.
2. Shore J, Warden S. *The Art of Agile Development*. O'Reilly Media, Inc., 2007.
3. Cooper K, Franch X. APLE first international workshop on agile product line engineering. SPLC '06. IEEE Computer Society: Silver Spring, MD, 2006; 205–206.

[3] Application Engineering process [1]

Generación de Documentos con Contenido Variable en DPLfw*

Abel Gómez, Mª Carmen Penadés, and José H. Canós

ISSI – DSIC, Universitat Politècnica de València.
Cno. de Vera, s/n. 46022 Valencia. Spain.
{agomez,mpenades,jhcanos}@dsic.upv.es

Resumen Actualmente existen soluciones tecnológicas para la generación de documentos personalizados en cuanto a sus contenidos y apariencia. Sin embargo, todas ellas requieren de amplios conocimientos en lenguajes especializados (XML, XSLT o XPATH entre otros) y no contemplan tareas relacionadas específicamente con el dominio, como es la identificación de la variabilidad en el contenido de los documentos. En este trabajo presentamos DPLfw, un entorno de trabajo basado en modelos para la generación de documentos con contenido variable. DPLfw es una implementación de la propuesta de Líneas de Producto de Documentos, donde la variabilidad en el contenido se representa mediante características, y la generación de documentos se soporta sobre un proceso basado en Líneas de Productos. Este artículo describe la arquitectura de DPLfw, a la vez que muestra su uso en la generación de documentación de usuario.

Palabras Clave: Impresión de Datos Variables, Líneas de Productos de Documentos, Modelado de Características, Ingeniería Dirigida por Modelos, DITA

1. Introducción

Gestionar documentos con contenido variable es un aspecto fundamental en diversos ámbitos, tales como la enseñanza a distancia, el comercio electrónico o el gobierno electrónico. Los principales retos en la generación de manuales personalizados, contratos o documentos gubernamentales, por ejemplo, son la definición de variantes del documento y la reutilización de contenidos. En el campo de la Ingeniería de Documentos, la generación de documentos personalizados se conoce como Impresión de Datos Variables —Variable Data Printing (VDP)—. En las últimas décadas se han realizado diversas propuestas, desde las primeras al estilo de *Mail Merge* se ha llegado a propuestas con mayor grado de sofisticación, permitiendo la generación de documentos multimedia personalizados [13, 20]. Sin embargo, estas herramientas proporcionan una gestión de la variabilidad a bajo nivel donde el diseñador de documentos está obligado a tener un amplio

* Este artículo es una versión extendida de [7] preparada conforme a las recomendaciones de la llamada a contribuciones de JISBD 2012.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 629-642, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

conocimiento de XML y tecnologías asociadas para poder definir el documento personalizado. Además, en numerosas ocasiones, la reutilización es oportunista, en vez de potenciar una reutilización sistemática. En este contexto el objetivo es proporcionar una alternativa para la generación automática de documentos, a la vez que oculte la complejidad intrínseca a una alta variabilidad.

Una solución para ocultar la complejidad consiste en el desarrollo de herramientas para usuarios finales que acerquen la definición de la variabilidad al dominio del problema: los usuarios seleccionan qué contenidos de un documento son fijos y cuáles pueden variar independientemente del formato final del mismo. Además, la funcionalidad de dichas herramientas debe soportarse por una metodología que guíe el proceso de generación de un documento. Igualmente, y con el fin de incrementar la eficiencia, es deseable que se dé un soporte integrado para la reutilización de componentes de documento de forma sistemática.

Este artículo describe DPLfw, un *framework* para la generación de documentos con contenido variable que cumple con los requisitos anteriores. Este framework, presentado inicialmente en [7], está basado en las Líneas de Producto de Documentos —Document Product Lines (DPL) [17]— que proporcionan un marco de trabajo alternativo a la aproximación tradicional seguida en VDP. DPL permite la creación de documentos con contenido variable a usuarios no expertos y asegura la reutilización de contenidos a nivel del dominio, siguiendo los principios de la Ingeniería de Líneas de Productos Software —Software Product Line Engineering (SPLE) [4]—. La clave para el éxito de un proceso de DPL reside, por una parte, en la definición de un modelo de variabilidad (llamado modelo de características) que describa cómo pueden variar los documentos; y por otra parte, en la existencia de una colección organizada de componentes de documento (*core assets* según la terminología de SPLE). Los componentes de documento son piezas de contenido que pueden combinarse para producir el documento final de acuerdo con el modelo de características. DPLfw implementa el proceso DPL siguiendo los principios de la Ingeniería Dirigida por Modelos —Model Driven Engineering (MDE) [12]—, ilustrándose su funcionalidad en el caso de estudio presentado.

El resto del artículo se estructura como sigue. La sección 2 introduce DPL, mostrando cómo se han adaptado las técnicas de SPLE a la generación de documentos. La sección 3 describe el *framework* DPLfw. La sección 4 ilustra la generación de documentación de usuario. La sección 5 resume diversos trabajos relacionados. Finalmente se presentan las conclusiones y trabajos futuros.

2. Líneas de Producto de Documentos

DPL proporciona una guía metodológica para modelar los contenidos comunes y variables en familias de documentos como un conjunto de características. A partir de una selección de características para un documento concreto, se genera un editor personalizado (re)utilizando componentes, y siguiendo una aproximación de líneas de producto. Este editor es un artefacto que permite guiar el flujo de edición actuando de forma similar a un asistente. Su ejecución produce instancias

específicas de un documento con contenido variable que serán posteriormente visualizadas con una determinada presentación para producir la versión final del mismo. A continuación resumimos la metodología DPL.

2.1. Modelado de Características en Documentos

Un aspecto fundamental en DPL es la identificación de la variabilidad en documentos desde una perspectiva del dominio. Para soportar esta tarea, DPL adapta los modelos de características clásicos (notación FODA [10]) al dominio de la generación de documentos, teniendo en cuenta la visión general existente en la comunidad de Ingeniería de Documentos. De acuerdo con ella, un documento en DPL se define como la unión de contenido y presentación. Tomando por ejemplo los documentos para el pago de impuestos, encontramos que algunos contenidos son obligatorios, mientras que otros sólo son aplicables a casos específicos, pudiendo omitirse (variabilidad en el contenido). Por otra parte, los contenidos pueden mostrarse de diversas formas: por ejemplo, algunos pueden presentarse impresos, mientras que otros pueden presentarse mediante formularios electrónicos. Esta dimensión tecnológica se maneja en DPL de forma explícita.

Por lo tanto, en DPL se pueden tratar dos fuentes de variabilidad: contenido y tecnología, lo cual motiva distinguir dos tipos de características: aquellas relacionadas con el contenido del documento (*ContentDocumentFeature* o CDF); y aquellas relacionadas con la tecnología usada para representar dichos contenidos (*TechnologyDocumentFeature* o TDF). Una CDF puede asociarse a una o más TDFs. Además, se pueden definir relaciones cruzadas —como *requiere*, *excluye*, y combinaciones lógicas de ellas ($\wedge$ y $\vee$)— para expresar restricciones, como es habitual en el ámbito del modelado de características.

2.2. Proceso de Generación de un Documento

Un proceso DPL, al igual que en SPLE y según se describe detalladamente en [7], incluye dos subprocesos incrementales e iterativos. El primero, llamado *Ingeniería del Dominio* se compone de cuatro tareas. En la primera tarea, *Analizar la Familia de Documentos*, un ingeniero del dominio especifica la familia de documentos en términos de características de contenido y tecnología, produciendo un *modelo de características*. En la segunda tarea, *Diseñar la Familia de Documentos*, se define una arquitectura de referencia del documento, identificando los componentes de documento (relacionados con el contenido) y los componentes software (relacionados con la tecnología que soportan el contenido) que son necesarios según el modelo de características definido previamente.

En la tarea *Desarrollar Core-Assets* se identifican, se buscan y/o se desarrollan (en caso de que no existan) los componentes (de documento o software) necesarios para generar la línea de documentos, denominados genéricamente *core-assets*. Todos ellos se almacenan en un repositorio, y para su búsqueda y organización se hace uso de los metadatos asociados a cada uno de ellos. Finalmente, en la tarea *Generar Línea de Documentos* se diseña y obtiene un *plan de producción*, esto es, un proceso que especifica cómo se integran los diferentes

componentes de acuerdo a las relaciones definidas entre las características de la familia de documentos.

El segundo subproceso, *Ingeniería de la Aplicación*, da soporte al proceso específico de generación de documentos de contenido variable. En la tarea *Caracterizar Documento*, el ingeniero de documentos (encargado de generar un documento concreto) selecciona los puntos de variabilidad deseados, esto es, qué características opcionales y/o alternativas se incluirán en el documento. A continuación, en la tarea *Recuperar Core-Assets* se obtienen los *core-assets* según la selección realizada; y en la tarea *Generar Flujo de Creación del Documento*, los diferentes *core-assets* son ensamblados para construir un *Editor de Documento Personalizado*. Se incluyen tanto los de tecnología (componentes software capaces de mostrar un contenido) como los de contenido (contenido del documento que es conocido y muestra el editor *a priori*). Finalmente, en la tarea *Ejecutar Flujo de Creación del Documento*, se usa el editor para completar el contenido final (si es necesario), generándose el documento final.

3. El *Framework* DPLfw

DPLfw proporciona la base metodológica y tecnológica para crear documentos de contenido variable según DPL. DPLfw se ha desarrollado siguiendo los principios de la Ingeniería Dirigida por Modelos, que elevan el nivel de abstracción y reducen el esfuerzo necesario en el desarrollo mediante técnicas programación generativa. Además, DPLfw se ha diseñado para ser extensible y configurable.

Las principales tecnologías que dan soporte a la herramienta son dos: Equinox [14] y *Eclipse Modeling Framework*(EMF) [21]. Equinox es la implementación del proyecto Eclipse para estándar OSGi [16], un modelo dinámico de componentes así como una plataforma de servicios para construir aplicaciones Java modulares y extensibles. Por su parte, EMF es un marco de trabajo para construir aplicaciones basada en técnicas de MDE. DPLfw supone un paso más allá en la implementación de DPL con respecto a trabajos previos [17, 18], donde en lugar de usar una herramienta específica, se usaban plataformas de SPLE genéricas. Además, la herramienta actual proporciona mecanismos de verificación (gracias a el uso de FaMa Tool Suite [8]) y validación avanzados, mejorando la propuesta inicial de DPLfw realizada en [7].

La figura 1 muestra los principales elementos de DPLfw. El *Editor de Características* se usa para definir la variabilidad del dominio en un modelo de características de documento. Éste se comunica con el módulo de verificación de FaMa que ayuda al usuario en la definición del modelo. El *Editor de Componentes* permite crear nuevos *core-assets* que pueden almacenarse en el *Repositorio*. Los elementos citados son los que dan soporte a la fase de *Ingeniería del Dominio*, mientras que el resto dan soporte a la fase de *Ingeniería de la Aplicación*. En concreto, el *Editor de Configuraciones* permite la selección de los diferentes puntos de variabilidad, obteniendo una *configuración*. Gracias el módulo de validación se garantiza que la configuración definida por el usuario es válida y no viola ninguna de las restricciones definidas en el modelo de características. Dicha

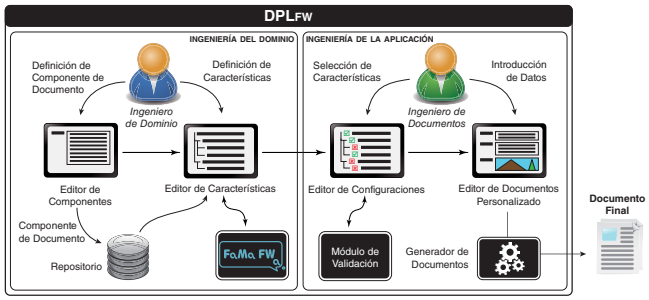


Figura 1: Vista general de DPLFW

configuración se utiliza para generar un *Editor de Documentos Personalizado*, que permite completar cualquier dato variable que pueda quedar por rellenar. Finalmente, el *Generador de Documentos* construye un documento final en un formato concreto (impreso, hipermedia, etc.).

3.1. El Repositorio

El proceso de automatización en la generación de documentos depende en gran medida en la disponibilidad de diferentes componentes que puedan ser reutilizados sistemáticamente. En DPLFW esta disponibilidad está garantizada por el *Repositorio*, un recurso que permite gestionar (crear, actualizar y borrar) y recuperar los distintos componentes (por ejemplo mediante búsqueda de palabras clave), siguiendo una arquitectura cliente/servidor. Se ha desarrollado siguiendo la filosofía de MDE y empleando la tecnología de EMF, por lo que éste es accedido vía *Connected Data Objects*(CDO) [6]. CDO proporciona un acceso de alto nivel (basado en modelos) para las aplicaciones, ofreciendo mecanismos de autenticación, acceso concurrente, transaccionalidad, así como de recuperación y almacenamiento de componentes independientemente del sistema de base de datos usado. A continuación, y por motivos de claridad y espacio centraremos el discurso en los componentes de contenido.

Los servicios proporcionados por el *Repositorio* dependen en gran medida de la estructura de los componentes de documento. En DPL, estos componentes de documento se denominan *InfoElements* [17]. Un *InfoElement* se compone de dos bloques principalmente: datos y metadatos. El primero es una codificación del contenido real del componente de documento (por ejemplo, texto, imagen, o cualquier otro objeto multimedia). El segundo permite proporcionar información adicional que permite describir y gestionar el bloque de contenido. El sistema de metadatos elegido en DPL está basado en el estándar *Dublin Core*. Ejemplos de metadatos son *Título*, *Descripción*, *Idioma*, *Creador*, *Materia*, *Tipo*, etc.

Dada esta descripción, hemos modelado el *Repositorio* en DPLFW según muestra la figura 2. Un *Repositorio* es accesible mediante una localización representada por un identificador único (URI), y contiene un conjunto de nodos (*resourceNodes*) que permiten organizarlo de forma jerárquica. Existen dos tipos

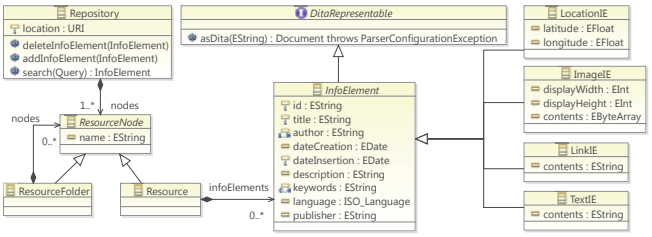


Figura 2: Modelo del *Repositorio*

de nodos: recursos (*Resources*) y carpetas(*ResourceFolders*). Los recursos pueden contener *InfoElements*, mientras que las carpetas sólo pueden contener otras carpetas o recursos. Los atributos de los *InfoElements* permiten representar los metadados. Por simplicidad, el metadato *Tipo*, que determina la codificación del contenido, se ha implementado mediante herencia. La figura muestra distintos tipos de *InfoElements*: texto (*textIE*), imagen (*ImageIE*), enlaces (*LinkIE*) y coordenadas geográficas (*LocationIE*). Finalmente, los *InfoElements* implementan la interfaz *DitaRepresentable*, lo que permite mantener compatibilidad hacia atrás con trabajos previos [17, 18] a la vez que permiten el uso de herramientas automatizadas para la generación de documentos (por ejemplo, *Dita Open Toolkit* [1]), mediante una representación XML según el estándar DITA [15].

3.2. El Editor de Características

El *Editor de Características* ha sido generado de forma automática empleando las herramientas de EMF. La figura 3 muestra una versión simplificada del metamodelo soportado por DPLfw. En DPLfw, un modelo de características de documento (*DocumentFeatureModel*) se compone de un conjunto de características de documento (*DocumentFeature*), las cuales, según la sección 2 pueden referirse al contenido (*ContentDocumentFeature*) o a la tecnología (*TecnologyDocumentFeature*). Los tipos de características, así como las restricciones aplicables, son las comunes en el área de SPLE tal y como se indica en la sección 2.1.

El contenido real de un documento se asocia a las características de contenido mediante instancias de la clase *InfoElement*. Dichas características de contenido están asociadas a una o más características de tecnología que definen cómo se representarán. El artefacto encargado de representar el contenido, según la terminología de [9], es un diseminador (*Disseminator*). Ejemplos de diseminadores son: editores de texto, visores de imágenes, reproductores de vídeo o cualquier otra aplicación (por ejemplo, un servicio web).

Una familia de documentos es, por tanto, un modelo conforme al metamodelo descrito, y que se define mediante el editor de características. Los elementos de dicho modelo serán los que guíen la definición de la arquitectura de referencia del editor personalizado que se generará.

Cabe destacar que el *Editor de Características* proporciona capacidades avanzadas de validación y verificación, como se demuestra en la sección 4.1. Para ello

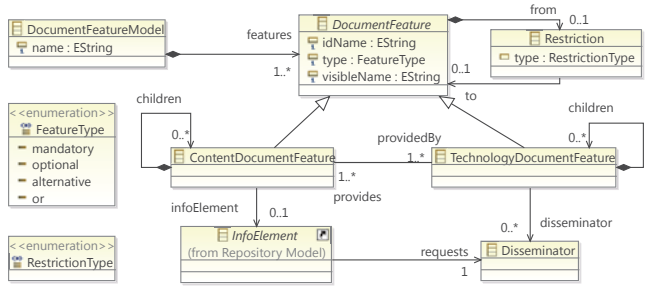


Figura 3: Metamodelo de Características de Documentos

se ha integrado de forma transparente para el usuario con FaMa Tool Suite [8], un entorno de trabajo para el análisis de modelos de características que se apoya en las representaciones lógicas y solucionadores más comunes en la literatura sobre modelado de características [2].

3.3. El Editor de Configuraciones

El *Editor de Configuraciones* permite guiar el proceso de caracterización (o selección de características) de un documento específico. Cada vez que el usuario realice una selección, se comprobará que todas las características obligatorias decidibles se seleccionen de forma automática; mientras que las características opcionales y alternativas deberán seleccionarse de forma explícita. Igualmente, cuando se deseleccione una característica, también se deseleccionarán todas sus características hijas. Por otra parte, todas aquellas características que no puedan seleccionarse serán marcadas como no seleccionables en el editor. Finalmente, en aquellos casos en los que se incumplan restricciones del modelo y no sea posible determinar de forma automática una solución (por ejemplo, en relaciones de dependencia o exclusión complejas), el editor mostrará en la vista de problemas aquellas relaciones conflictivas para que el usuario decida la acción correctiva, como se demuestra en la sección 4.2.

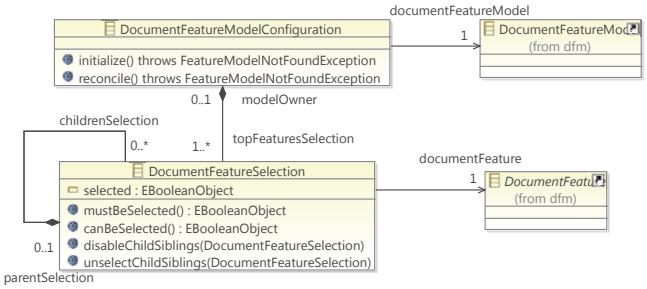


Figura 4: Metamodelo de Configuraciones de Documentos

Cada configuración de un modelo de características se almacena como un artefacto separado, cuyo contenido está enlazado al modelo de características asociado, manteniendo una estructura simple. Según la figura 4, una configuración de un modelo de características (*DocumentFeatureModelConfiguration*) está enlazado a un modelo de características (*DocumentFeatureModel*, véase la figura 3), y contiene una jerarquía de selecciones (*DocumentFeatureSelections*) cuya estructura se asemeja a la del modelo de características asociado. Una selección puede tener tres posibles valores: `true`, si la característica asociada está seleccionada; `false`, si la característica está deseleccionada de forma explícita; o `null` si el estado de la característica no se ha decidido todavía. En caso de que un modelo de características cambie, su modelo de configuración cambia automáticamente mediante un mecanismo de reconciliación. En caso de que, por ejemplo, se hayan añadido nuevas características que invaliden la configuración previa, los mecanismos de notificación de errores permiten guiar al usuario en las tareas de reconciliación que deben realizarse necesariamente de forma manual por no disponer de información suficiente.

3.4. Editor de Documentos Personalizado y Generación del Documento Final

El *Editor de Documentos Personalizado* implementa el *flujo de creación del documento*, permitiendo especificar por completo el documento final antes de ser generado. Siguiendo el ejemplo previo sobre documentos para el pago de impuestos (sección 2.1), el flujo de creación del mismo especifica qué contenidos deben rellenarse y en qué orden. Este editor personalizado se genera a partir de la configuración definida. Finalmente, y una vez que los contenidos han sido rellenado por completo, el *Generador de Documentos* construye el documento final con su contenido y visualización definitivos.

DPLfw, en su estado actual de desarrollo, no soporta por completo la generación de editores personalizados. Sin embargo, para cubrir esta fase de DPL, se han implementado diversos mecanismos que proporcionan un comportamiento por defecto. Esto permite llevar a cabo las tareas de *Generación del Flujo de Creación del Documento* y *Ejecución del Flujo de Creación del Documento* empleando herramientas alternativas. En concreto, puesto que los *InfoElements* pueden representarse como especificaciones DITA, una configuración de un documento se transforma de forma automática a un *mapa DITA*, que representa el flujo de edición del documento. Este *mapa DITA* puede editarse empleando el editor integrado en DPLfw (que juega el papel del *Editor de Documentos Personalizado*) y se usa posteriormente para generar el documento final empleando el motor *DITA Open Toolkit* [1]. De esta manera, DPLfw cubre un proceso DPL por completo, permitiendo la edición y generación de cualquier documento con contenido variable, pudiendo ser su formato final un archivo PDF, RTF, un conjunto de documentos HTML enlazados, etc.

4. Caso de estudio: Generación de un Manual de Usuario

La generación de documentos con contenido variable pretende cambiar la filosofía de «1 documento para N personas» a «1 documento para 1 persona». En el caso de documentación de aplicaciones software, puede ser relevante disponer de distintos manuales que documenten distintos aspectos de una herramienta, por ejemplo: requisitos e instrucciones de instalación para los administradores de sistemas; guía de primeros pasos y manual de uso para usuarios finales; combinaciones de ambos para usuarios avanzados; etc. A lo largo de esta sección mostraremos cómo DPLfw ha soportado la generación de documentación para la propia herramienta.

4.1. Ingeniería del dominio

La figura 5 muestra el modelo de características que describe la variabilidad de la familia de manuales de la herramienta DPLfw. En el modelo encontramos cuatro características de contenido (CDF) de primer nivel: (1) *Introducción*, que realiza una introducción a la herramienta dependiendo de dos tipos de usuarios distintos (administradores de sistemas o usuarios finales); (2) *Instrucciones de Instalación*, que describe los requisitos de la herramienta y los pasos para instalar tanto el servidor (el repositorio) como el cliente de DPLfw; (3) la guía de *Primeros Pasos*, que enseña a los usuarios cómo empezar a trabajar rápidamente con la herramienta; y (4) el *Información de Versiones*, que describe la evolución de la herramienta a lo largo de sus distintas versiones. Por simplicidad el modelo sólo muestra una única característica de tecnología (TDF), el *Tipo de Manual*, que puede ser como documento impreso o como documento multimedia. Igualmente, se han definido diferentes restricciones de dependencia y exclusión. Por ejemplo, si se selecciona una introducción para *Administradores de Sistemas* se requiere que posteriormente se incluyan las instrucciones de instalación del servidor, así como la información de versiones. Por su parte si se realiza una introducción para usuarios finales, se requiere incluir la guía de primeros pasos.

Para ilustrar las capacidades de análisis de DPLfw se han introducido además dos relaciones de exclusión mutua (que invalidan el modelo) entre la información de las versiones *v0.2.1* y *v0.3*. De esta manera, se puede observar el resultado del análisis en la vista de problemas (figura 5, derecha). Específicamente, esta vista indica que existen *características muertas* (nunca pueden seleccionarse sin violar las restricciones del modelo). Esto se debe a que las características *v0.2.1* y *v0.3* son obligatorias en caso de que *Información de versiones* esté seleccionada, sin embargo al ser estas dos últimas mutuamente excluyentes, no pueden seleccionarse ninguna de las tres. A su vez, la característica opcional *Administración de sistemas* tampoco puede ser seleccionada nunca puesto que requiere *Información de versiones*, que como se ha visto, es una característica muerta. Por último, el editor también indica que *Primeros pasos* es una característica *obligatoria falsa*, es decir, a pesar de estar declarada como opcional, todas las configuraciones posibles la contienen, comportándose como si fuera obligato-

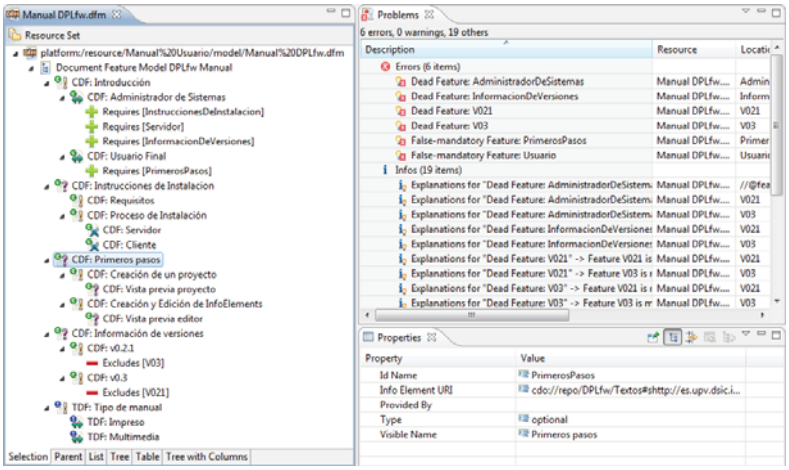


Figura 5: Modelo de características de ejemplo (con errores)

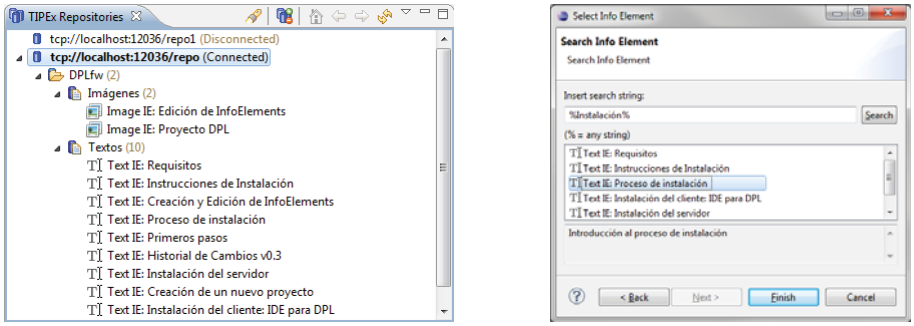
ria. Para obtener un modelo de características válido basta con eliminar las dos relaciones de exclusión.

A la vez que se define la familia de documentos se deben desarrollar/recuperar los *core-assets* que se van a asociar a cada característica mediante el editor mostrado en la figura 5. Para añadir, eliminar y gestionar los diferentes componentes de documento se dispone del *Explorador de repositorios* mostrado en la figura 6a. Como se observa, en un repositorio los componentes se organizan de forma jerárquica. Sin embargo, para la búsqueda y recuperación de componentes se dispone de una interfaz de búsqueda que facilita aún más la tarea, tal y como muestra la figura 6b. Para la creación y edición de componentes de documento DPLfw proporciona distintos editores, que, empleando una metáfora de formulario, permiten rellenar tanto los metadatos como el contenido de éstos.

El último paso de la *Ingeniería del Dominio* es la generación de la línea de productos de documento, obteniendo el plan de producción. DPLfw implementa un plan de producción por defecto donde: la estructura de la familia de documentos está determinada por el modelo de características; cada componente de contenido (*InfoElement*) tiene un diseminador por defecto; y los mecanismos para recuperar los diferentes componentes del repositorio están predefinidos.

4.2. Ingeniería de la aplicación

En la fase de *Ingeniería de la Aplicación*, el ingeniero de documentos emplea los modelos de variabilidad definidos previamente para generar el documento final. La figura 7a muestra el *Editor de Configuraciones* mientras se está definiendo un documento dirigido a un administrador de sistemas, que será el encargado de implantar DPLfw en una organización. Se puede observar cómo la configuración está parcialmente definida, siendo ésta incorrecta. La vista de problemas informa



(a) (b)

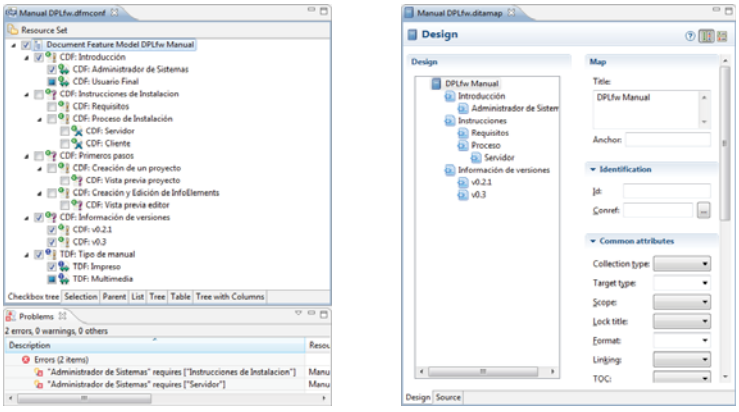
Figura 6: Explorador de repositorios (6a) y Buscador de InfoElements (6b)

de que al seleccionar una introducción para administradores de sistemas, también es necesario incluir las instrucciones de instalación del servidor de DPLfw según se definía en el modelo mostrado en la figura 5. En la figura también se observa que se ha seleccionado la característica de tecnología *tipo de manual impreso*, que nos permitirá generar un documento PDF.

Tras seleccionar las características *Instrucciones de Instalación*, y *Servidor* (*Requisitos* y *Proceso de Instalación* se seleccionan de forma automática al ser obligatorias), ya se dispone de una selección válida y el proceso continúa. Partiendo de esta caracterización se puede generar un documento acorde a ella. Para ello, en primer lugar, se recuperan los componentes de documento de los respectivos repositorios y se almacenan localmente como especificaciones DITA; y en segundo lugar, se genera automáticamente un mapa DITA que describe la estructura del documento. Todos estos elementos pueden editarse con los correspondientes editores que DPLfw integra. Por ejemplo, la figura 7b muestra el editor de mapas DITA correspondiente al caso de estudio. Por último, el generador construye el documento final. La figura 7c muestra el documento PDF que se genera para la selección de características descrita anteriormente.

5. Trabajos relacionados

El interés en los documentos de contenido variable ha crecido de forma significativa en los últimos años, especialmente en la comunidad de Ingeniería de Documentos. DARWIN VDP [5] y PageFlex [3] son ejemplos de soluciones tecnológicas que proporcionan mecanismos para la personalización de la presentación así como instrucciones avanzadas para el procesado de datos. Otros autores han abordado el problema mediante la edición de documentos XML, por ejemplo [13, 20]. Las herramientas están orientadas a la presentación, centrándose en el documento final y cómo las partes variables del documento se instancian. Sin embargo, no proporcionan ninguna guía metodológica para identificar y gestionar de forma temprana dichas fuentes de variabilidad, aspecto que es esencial



(a) (b)

(c)

Figura 7: Configuración (con errores) (7a), mapa DITA generado tras la corrección (7b y manual de usuario final en PDF (7c)

en dominios complejos. Además, su orientación puramente tecnológica y de bajo nivel les impide gestionar de forma genérica la variabilidad tecnológica.

Un caso interesante de gestión temprana de la variabilidad en documento se describe en [19]. En este trabajo se aborda la problemática de alinear el contenido de los documentos con las variantes de una línea de productos empleando DOPLER, un entorno para el desarrollo de SPLs orientadas por decisiones. En este trabajo se presenta, además, una metodología para llevar a cabo el proceso de generación de documentos. Sin embargo a diferencia de DPLfw, el principal objetivo de esta propuesta no es la generación de documentos, entendiendo estos como los artefactos resultantes de una línea de productos. Así, la propuesta más próxima a DPL la encontramos en [11], que utiliza FODA para especificar la variabilidad en documentos. Así, las características son asociadas a documentos específicos, y mediante un mecanismo de asignación se construye una determinada variante a partir de una selección de características. A diferencia de DPL, no existe ninguna distinción entre características de contenido y tecnología. Además, esta propuesta no sigue una aproximación de SPLE pura, sino que se utiliza una

aproximación transformacional. Finalmente, su uso está dirigido a aplicaciones de oficina, mientras que DPL es una herramienta de propósito general.

6. Conclusiones y trabajo futuro

La generación de documentos personalizados con reutilización de contenido es una cuestión fundamental en numerosos ámbitos. Para abordar este problema hemos presentado DPLfw, un *framework* y una herramienta que dan soporte a la metodología DPL para la generación de documentos basada en reutilización. Un proceso DPL empieza con la definición de un modelo que define las características de una familia de documentos, y a partir del cual se deriva un arquitectura de documento genérica. Así, un documento concreto (un miembro de la familia) se crea siguiendo un proceso donde se obtienen diferentes componentes de un repositorio y se ensamblan posteriormente, maximizando la reutilización. DPL está basada en los principios de SPLE, capturando la variabilidad en etapas tempranas, a diferencia de otras aproximaciones que están orientadas a la presentación y la tecnología de generación del documento final.

DPLfw es una herramienta completamente funcional que cubre por completo el ciclo de vida de un documento, desde la fase de análisis del dominio (donde se especifica la familia de documentos), hasta la obtención de un documento final. DPLfw mejora trabajos previos [7] y permite además modelar de forma sencilla modelos de características complejos: en primer lugar, permite garantizar que los modelos no contienen errores en la fase de modelado gracias a herramientas como FaMa; en segundo lugar, asiste al usuario en la fase de caracterización mediante selecciones automáticas y avisos; y en tercer lugar, permite la edición concurrente de modelos de características y configuraciones que se sincronizan mediante mecanismos automáticos de reconciliación.

Además del caso de estudio empleado para ilustrar la herramienta, DPLfw ha sido utilizado en casos de estudio reales, especialmente en la generación de los llamados *Planes de Emergencia* (documentos que sirven de guía de seguridad en el caso de situaciones de riesgo). Finalmente, cabe destacar que a pesar de que DPLfw es completamente funcional, aún resta por completar la construcción de editores para ejecutar el flujo de edición que permitirían explotar las capacidades de DPLfw al máximo. En este sentido, DPLfw ya soporta la implementación y reutilización de distintos tipos de diseminadores, restando únicamente la composición de éstos para la generación de editores personalizados.

Agradecimientos Este trabajo está parcialmente financiado por el Ministerio de Educación y Ciencia bajo el proyecto TIPEx (ref. TIN2010-19859-C03-03).

Referencias

[1] Anderson, Robert D and Shen, Jian Le and Elovirta, Jarno and Sirois, Eric and Weiser, Reuven: DITA Open Toolkit (2012), `http://dita-ot.sourceforge.net/`

[2] Benavides, D., Segura, S., Ruiz-Cortés, A.: Automated analysis of feature models 20 years later: A literature review. Information Systems 35(6), 615 – 636 (2010)
[3] Bitstream Inc.: Pageflex (2012), http://www.pageflex.com/
[4] Clements, P., Northrop, L.: Software Product Lines: practices and patterns, vol. 0201703327. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2001)
[5] CREO: DARWIN VDP (2012), http://www.creo.com/data/Products/Workflow%20Solutions/Darwin%20VI%20authoring%20tool/
[6] Eclipse Foundation: CDO (2012), http://www.eclipse.org/cdo/
[7] Gómez, A., Penadés, M.C., Canós, J.H., Borges, M.R., Llavador, M.: DPLFW: A framework for Variable Content Document Generation. In: 16th International Software Product Line Conference, SPLC 2012, Salvador, Brazil, September 2-7 (2012)
[8] ISA Research Group: FaMa FW (2012), http://www.isa.us.es/fama/
[9] Kahn, R., Wilensky, R.: A framework for distributed digital object services. Int. J. Digit. Libr. 6, 115–123 (April 2006)
[10] Kang, K., Cohen, S., Hess, J., Nowak, W., Peterson, S.: Feature-Oriented Domain Analysis (FODA) Feasibility Study (1990)
[11] Karol, S., Heinzerling, M., Heidenreich, F., Assmann, U.: Using feature models for creating families of documents. In: DocEng '10. pp. 259–262. ACM, New York, NY, USA (2010)
[12] Kent, S.: Model driven engineering. In: Butler, M.J., Petre, L., Sere, K. (eds.) IFM 2002, Finland. LNCS, vol. 2335, pp. 286–298. Springer (2002)
[13] Lumley, J., Gimson, R., Rees, O.: A framework for structure, layout & function in documents. In: DocEng '05. pp. 32–41. ACM (2005)
[14] McAffer, J., VanderLei, P., Archer, S.: OSGi and Equinox: Creating Highly Modular Java Systems. Eclipse series, Addison-Wesley (2009)
[15] OASIS: Darwin Information Typing Architecture (DITA) v.1.2. Organization for the Advancement of Structured Information Standards (Dec 2010), http://docs.oasis-open.org/dita/v1.2/spec/DITA1.2-spec.html
[16] OSGi Alliance: OSGi Service Platform Core Specification. OSGi Alliance (2008), http://www.osgi.org/Specifications/HomePage/
[17] Penadés, M.C., Canós, J.H., Borges, M.R., Llavador, M.: Document product lines: variability-driven document generation. In: DocEng '10. pp. 203–206. ACM (2010)
[18] Penadés, M.C., Canós, J.H., Camarasa, S., Borges, M.R., Vivacqua, A.S.: Generación de documentos basada en líneas de producto. JISBD'11. Spain (2011)
[19] Rabiser, R., Heider, W., Elsner, C., Lehofer, M., Grünbacher, P., Schwanninger, C.: A flexible approach for generating product-specific documents in product lines. In: Bosch, J., Lee, J. (eds.) SPLC. LNCS, vol. 6287, pp. 47–61. Springer (2010)
[20] Sellman, R.: VDP templates with theme-driven layer variants. In: DocEng'07. pp. 53–55. ACM (2007)
[21] Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: EMF: Eclipse Modeling Framework. Addison-Wesley, Boston, MA, 2nd edition edn. (2009)

BeTTy: Un Framework de Pruebas para el Análisis Automático de Modelos de Características

Sergio Segura, José A. Galindo, David Benavides and José A. Parejo

Departamento de Lenguajes y Sistemas Informáticos. Universidad de Sevilla.
Av Reina Mercedes S/N, 41012 Sevilla, España

Resumen El análisis automático de modelos de características es un área de investigación activo que ha llamado la atención de numerosos investigadores durante las dos últimas décadas. Durante este tiempo, el número de herramientas y técnicas para el análisis de modelos de características se ha multiplicado y con ellas su complejidad. En este escenario, la falta de mecanismos específicos para validar y evaluar la funcionalidad y el rendimiento de las herramientas de análisis se ha convertido en un gran obstáculo dificultando el desarrollo de herramientas y afectando negativamente a su calidad y fiabilidad. En este artículo, presentamos BeTTy, un framework para la automatización de pruebas en el análisis de modelos de características. Entre otras funcionalidades, BeTTy permite la detección automática de errores en herramientas de análisis de modelos de características. Además, BeTTy permite generar modelos de características tanto aleatorios como computacionalmente duros, útiles para evaluar el rendimiento de las herramientas de análisis. Parte de la funcionalidad del framework es ofrecida a través de una aplicación Web que facilita en gran medida su uso.

1. El Framework BeTTy

La ingeniería de líneas de productos es un paradigma de desarrollo orientado a construir familias de sistemas software que reutilicen características comunes en lugar de construir cada producto desde cero. Un aspecto fundamental para la gestión de una línea de productos es utilizar un modelo que permita representar todos los posibles productos que pueden derivarse de ella. Uno de los modelos más populares usados para tal fin son los denominados modelos de características [3]. Un *modelo de características* se representa visualmente como un árbol donde los nodos representan las características y las aristas representan las posibles relaciones o restricciones entre características.

La extracción automática de información de modelos de características es un tema de investigación activo que ha atraído la atención de numerosos investigadores en los últimos veinte años [1]. Durante este tiempo, se han presentado un gran número de operaciones, técnicas y herramientas y se ha consolidado toda una comunidad alrededor de lo que se ha denominado *análisis automático*

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 643-646, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

de modelos de características. Actualmente, los avances en éste área están llevando a una mayor preocupación por la calidad de las herramientas de análisis. Los prototipos de investigación ya no son suficiente y ahora se buscan solidas herramientas de análisis en términos de ausencia de errores y rendimiento. Sin embargo, las pruebas software empleadas en este campo son fundamentalmente aleatorias y guiadas por la intuición de los propios desarrolladores más que por técnicas maduras para el diseño de datos de prueba. Esto hace que las conclusiones de las pruebas sean difícilmente rigurosas y verificables a la vez que limitan el alcance y el valor de las contribuciones científicas en el análisis de modelos de características. De igual forma, la arbitrariedad u omisión de las pruebas en este campo reducen notablemente la confianza de los usuarios sobre el correcto funcionamiento de las herramientas de análisis.

Para abordar el problema de las pruebas en el análisis de modelos de características, hemos presentado una serie de técnicas, algoritmos y herramientas para la realización de pruebas funcionales y de rendimiento en herramientas de análisis de modelos de características. Estas contribuciones son el resultado de la aplicación de varias técnicas clásicas e innovadoras de pruebas software en el contexto de análisis de modelos de características. Para facilitar las pruebas funcionales, hemos presentado un conjunto de casos de prueba [4] así como un generador automático de datos de prueba para la detección de errores en las herramientas de análisis [6,7]. En lo que se refiere a pruebas de rendimiento, hemos trabajado en un algoritmo evolutivo para la generación de modelos de características difíciles de procesar en términos de tiempo y memoria requeridos para su análisis [8]. Estas herramientas han sido evaluadas experimentalmente de forma rigurosa demostrando la viabilidad de la propuesta. Entre otros resultados, nuestras herramientas nos han permitido detectar dos errores en FaMa [2] y otros dos en SPLOT [9], dos herramientas para el análisis de modelos de características ampliamente usadas por la comunidad. Con el objetivo de hacer nuestras contribuciones accesibles a la comunidad y animar a investigadores y profesionales a usar, extender y evaluar nuestro trabajo, hemos integrado todas estas contribuciones en un framework llamado BeTTy (Benchmarking and TesTing on the analYsis of feature models). Las principales funcionalidades del framework son:

- **Generación de modelos de características aleatorios**. BeTTy permite la generación aleatoria de modelos de características. Estos pueden ser usados para evaluar el rendimiento de las herramientas de análisis con modelos de diversos tamaños y formas. Los modelos generados pueden almacenarse en hasta cinco formatos diferentes facilitando la interoperabilidad con otras herramientas.
- **Generación aleatoria de modelos de características atribuidos**. BeTTy también permite la generación de modelos de características atribuidos, también llamados modelos de características extendidos. Estos modelos son ampliamente usados para añadir atributos extra funcionales a las características. Hasta donde sabemos, esta es la primera herramienta disponible para la generación aleatoria de este tipo de modelos.

- **Generación automática de datos de prueba para pruebas funcionales**. BeTTy permite la generación de entradas y salidas esperadas para un gran número de operaciones de análisis sobre modelos de características (18 hasta el momento) facilitando la detección de errores en herramientas de análisis.
- **Generador evolutivo de modelos de características**. BeTTy incluye un algoritmo evolutivo para la generación de modelos de características que maximicen un criterio de optimización dado por el usuario. Esta característica puede ser usada, por ejemplo, para generar modelos de características computacionalmente duros (maximizando el tiempo de análisis) que permitan evaluar el rendimiento de las herramientas de análisis en casos pesimistas.
- **Benchmarking**. BeTTy integra varios componentes para facilitar la comparación del rendimiento de varias herramientas de análisis, esto es, generación de datos de prueba, ejecución y almacenamiento de resultados.

BeTTy está escrito en Java, liberado bajo licencia LGPL3 y distribuido como un fichero jar para facilitar su integración en proyectos externos. Además, parte de la funcionalidad del framework está disponible a través de una aplicación Web (ver Figura 1) permitiendo la generación de modelos de características aleatorios

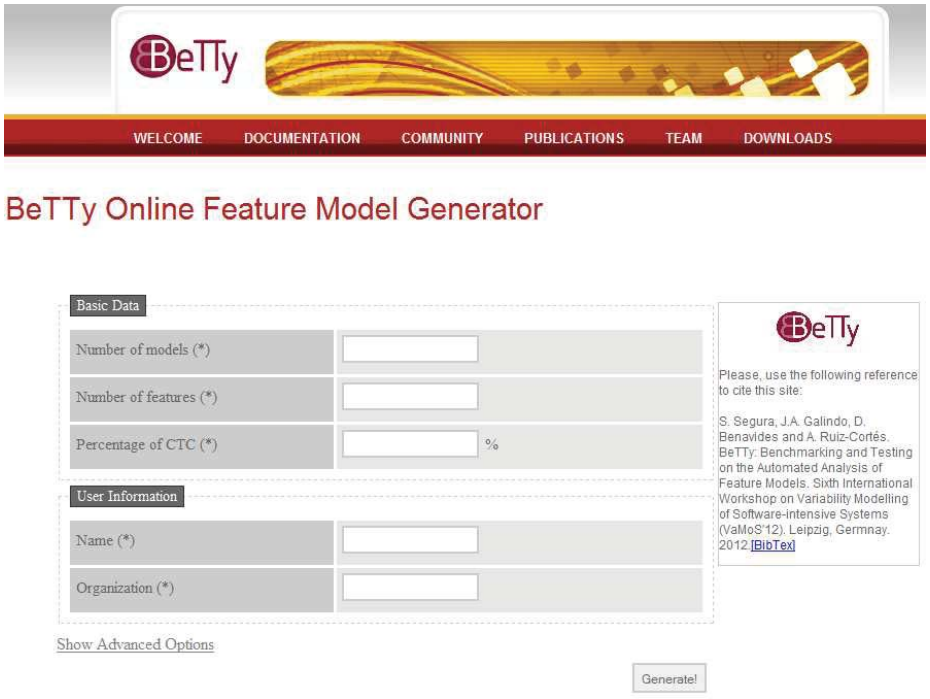


Figura 1. Generador on-line the modelos de características de la Web de BeTTy

con unos pocos clicks. Para más información sobre BeTTy el lector interesado puede consultar [5].

El framework puede descargarse desde la Web de BeTTy: `www.isa.us.es/betty`.

Agradecimientos

Este trabajo ha sido financiado parcialmente por la comisión europea y el gobierno español mediante el proyecto CICYT SETI (TIN2009-07366), y por el gobierno andaluz mediante los proyectos de excelencia ISABEL(TIC-2533) y THEOS (TIC-5906).

Referencias

1. D. Benavides, S. Segura, and A. Ruiz-Cortés. Automated analysis of feature models 20 years later: A literature review. *Information Systems*, 35(6):615 – 636, 2010.
2. FaMa Tool Suite. `http://www.isa.us.es/fama/`, accessed November 2011.
3. K. Kang, S. Cohen, J. Hess, W. Novak, and S. Peterson. Feature–Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21, SEI, 1990.
4. S. Segura, D. Benavides, and A. Ruiz-Cortés. Functional testing of feature model analysis tools: a test suite. *Software, IET*, 5(1):70 –82, february 2011.
5. S. Segura, J.A. Galindo, D. Benavides, J.A. Parejo, and A. Ruiz-Cortés. BeTTy: Benchmarking and testing on the automated analysis of feature models. In U.W. Eisenecker, S. Apel, and S. Gnesi, editors, *Sixth International Workshop on Variability Modelling of Software-intensive Systems (VaMoS'12)*, pages 63–71, Leipzig, Germany, 2012. ACM.
6. S. Segura, R.M. Hierons, D. Benavides, and A. Ruiz-Cortés. Automated test data generation on the analyses of feature models: A metamorphic testing approach. In *International Conference on Software Testing, Verification and Validation*, pages 35–44, Paris, France, 2010. IEEE press.
7. S. Segura, R.M. Hierons, D. Benavides, and A. Ruiz-Cortés. Automated metamorphic testing on the analyses of feature models. *Information and Software Technology*, 53(3):245 – 258, 2011.
8. S. Segura, JA. Parejo, Robert M. Hierons, D. Benavides, and A. Ruiz-Cortés. ETHOM: An evolutionary algorithm for optimized feature models generation. Tech Report ISA-2011-TR-03 (v. 1.0), Applied Software Engineering Research Group, 2011.
9. S.P.L.O.T.: Software Product Lines Online Tools. `http://www.splot-research.org/`, accessed November 2011.

A Systematic Review of Quality Attributes and Measures for Software Product Lines*

Silvia Abrahão, Sonia Montagud, Emilio Insfran

Grupo ISSI, Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València
Camino de Vera s/n, 46022, Valencia, España

{sabrahao,smontagud,einsfran}@dsic.upv.es

1 Introduction

While quality is an important factor in the construction of single software products, it becomes crucial in Software Product Lines (SPL) since the quality of all products that can be derived from the product line must be ensured. Software measures provide an appropriate mechanism for understanding, controlling, and predicting the quality of software development projects.

A great number of software measures for assessing the quality of Software Product Lines (SPL) have been proposed over the last few years. However, no studies summarizing the current knowledge about them exist. This paper presents a systematic literature review with the aim of identifying and analyzing the existing quality attributes and measures proposed by researchers from 1996 to 2010 to evaluate the quality of software product lines.

2 Planning the systematic review

To identify the primary studies for our systematic literature review, we used the following digital libraries: IEEEXplore, ACM Digital Library, Science Direct, SpringerLink, and INSPEC. Primary study is the common term to describe the publications contributing to a SLR. We tested several search strings, and the following one retrieved the greatest number of relevant papers: *((attribute* OR factor* OR propert* OR criterion OR criteria OR characteristic*) OR (metric* OR measur*)) AND (software OR engineering OR architectur*) AND ("product line*" OR "product famil*") AND (quality OR non-functional OR "no functional" OR assess* OR assur*)*. In addition, we performed a manual search in relevant conference proceedings.

To extract the data from the selected studies, the following criteria were used: (1) quality characteristic evaluated (those from the ISO/IEC 25010): (2) quality attribute evaluated by the measure (e.g., internal, external; specific for SPL); (3) type of measure (base, derived); (4) type of evaluation (e.g., qualitative, quantitative, probabilis-

* This work has been funded by the MULTIPLE project (TIN2009-13838)

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 647-648, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

tic); (5) phase of the SPL life cycle in which the measure is applied; (6) artifact(s) used to apply the measure; (7) other characteristics (e.g., compositionality, variability); (8) validation procedure (theoretical, empirical, not validated); (9) tool support (manual, automated), and (10) current usage (academic, industrial).

3 Results of the systematic review

The systematic review identified 173 quality attributes and 165 software measures. Of these quality attributes, 76 do not have any measure to quantify them. With regard to the measures, the most significant result is that only 25% of them were empirically validated. Therefore, there is a need for more validation (both theoretical and empirical) since validation is an essential process to ensure the usefulness of software measures. A further significant result is that a high number of measures were proposed for evaluating the maintainability characteristic (92%), whilst the remaining quality characteristics were drastically ignored.

With regard to the SPL life cycle in which the measures are applied, the review shows that the majority of measures were used to evaluate the quality of software artifacts produced during the requirements (9%) and the design phase (67%) of the Domain Engineering. This means that most of the measures focus on early phases of the SPL development. We believe that it is important to evaluate the quality in these phases since the SPL architecture and the core assets are developed during domain engineering but it is also important to ensure quality along the whole SPL life cycle.

4 Conclusions

The results obtained show that there is a need for new measures to cover different gaps (e.g., to quantify quality attributes related to other characteristics such as security, usability, safety and performance) and to properly validate the existing measures in order to provide more evidence about their usefulness.

These results may be useful to product line developers, acquirers, quality assurance staff and independent evaluators, particularly those responsible for specifying quality models and evaluating the quality of product line artifacts. In addition, our results may be useful as a reference guide for practitioners to assist them in the selection or the adaptation of existing measures for evaluating their software product lines.

References

1. Montagud S., Abrahão S., Insfran E.: A Systematic Review of Quality Attributes and Measures for Software Product Lines. Software Quality Journal, Special Issue on Quality Engineering for Software Product Lines, ISSN: 0963-9314, Springer (2011).

Sesión Temática 7

Miscelánea

Coordinadora: *Dra. Coral Calero*

Sesión Temática 7: Miscelánea
Coordinadora: Dra. Coral Calero

John W. Castro, Silvia T. Acuña, Oscar Dieste. *Diferencias entre las Actividades de Mantenimiento en los Procesos de Desarrollo Tradicional y Open Source*. (Regular)

María Fernández-Ropero, Ricardo Pérez-Castillo, Mario Piattini. *Refactorización selectiva de Procesos de Negocio*. (Regular)

José Luis Fernández-Alemán, et al. *Accessibility and Internationalization in Requirements Engineering Tools*. (Regular)

Gorka Guerrero, Roberto Yus, and Eduardo Mena. *Using Small Affordable Robots for Hybrid Simulation of Wireless Data Access Systems*. (Regular)

Pablo Ortíz, et al. *Agile Moodle: Una plataforma para el Aprendizaje Ágil en Ingeniería del Software*. (Herramienta)

M. Cruz, et al. *Auditoría de procesos de negocio en la nube: persistencia mediante almacenes no relacionales*. (Emergente)

A. Ruíz, L. Iribarne (Eds.): Actas de las "*XVII Jornadas de Ingeniería del Software y Bases de Datos (JISBD'2012)*", Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

Diferencias entre las Actividades de Mantenimiento en los Procesos de Desarrollo Tradicional y Open Source

John W. Castro[1], Silvia T. Acuña[1], Oscar Dieste[2]

[1]Departamento de Ingeniería Informática, Universidad Autónoma de Madrid
Calle Francisco Tomás y Valiente 11, 28049 Madrid, España
john.castro@estudiante.uam.es, silvia.acunna@uam.es
[2]Facultad de Informática, Universidad Politécnica de Madrid
Campus de Montegancedo s/n, 28660 Boadilla del Monte, Madrid, España
odieste@fi.upm.es

Resumen. ***Antecedentes***. La creciente importancia del *Open Source Software* (OSS) ha llevado a los investigadores a estudiar cómo los procesos OSS difieren de los procesos de la ingeniería del software tradicional. ***Objetivo***. Determinar las diferencias y similitudes entre las actividades del proceso de mantenimiento seguido por la comunidad OSS y el establecido por el estándar IEEE 1074:2006. ***Método***. Para conocer las actividades que conforman el proceso de desarrollo OSS realizamos un *Systematic Mapping Study*. Posteriormente, realizamos un emparejamiento entre las actividades del estándar IEEE 1074:2006 con las actividades del proceso OSS. ***Resultados***. Encontramos un total de 22 estudios primarios. De estos estudios, el 73% contaban con actividades relacionadas con el proceso de mantenimiento. ***Conclusiones***. El proceso de mantenimiento tradicional del software no encaja con lo que ocurre en la comunidad OSS. En su lugar, puede ser mejor caracterizar la dinámica general de la evolución OSS como reinvención. Esta reinvención emerge continuamente de la adaptación, aprendizaje, y mejora de las funcionalidades y calidad del OSS. Los proyectos OSS evolucionan a través de mejoras menores donde participan tanto usuarios como desarrolladores.

Palabras clave. *Systematic Mapping Study*, Proceso de Mantenimiento, *Open Source Software*.

1 Introducción

La creciente importancia del *Open Source Software* (OSS) en los últimos años ha llevado a los investigadores a estudiar cómo los procesos OSS difieren de los procesos de ingeniería del software tradicional. La comprensión del contexto, estructura y actividades de los procesos de desarrollo OSS que se encuentran en la práctica ha sido y sigue siendo un problema difícil de resolver [15].

Algunos estudios han demostrado que los procesos OSS son diferentes en muchos aspectos de los procesos software tradicionales [15][16][11]. Por otro lado, algunos autores como Fuggetta [4] y Godfrey y Tu [5] no son de tal opinión y afirman que el

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 651-664, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

modelo de desarrollo OSS no es exactamente una nueva descripción del proceso, sino sólo una visión alternativa de las actividades de ingeniería del software aplicadas a los modelos de desarrollo comercial tradicional. En cualquier caso, podemos fácilmente encontrar numerosos ejemplos de sistemas OSS exitosos, como los estudiados por Mockus et al. [8][9]. Por tanto, los procesos seguidos por la comunidad OSS objetivamente funcionan ya que obtienen resultados de calidad reconocidos a nivel mundial, lo cual a su vez implica que una mejor comprensión de los mismos puede informar y favorecer los procesos de desarrollo de software más habituales.

Existen pocos trabajos que analicen el proceso de desarrollo OSS. El trabajo de [3] realiza una revisión de la literatura relacionada con el proceso de desarrollo OSS, en particular sobre las prácticas seguidas por los equipos de desarrollo OSS durante el análisis de requisitos, la implementación, las pruebas, la liberación de versiones y el mantenimiento. El proceso de requisitos seguido en proyectos de desarrollo OSS en diferentes dominios (como la astrofísica, juegos de ordenador en red y sistemas de diseño de software) ha sido estudiado por Scacchi y colegas [12][13][14]. En estos estudios se ha encontrado, por ejemplo, que por lo general no hay documentos de requisitos de software explícitos. Los procesos de requisitos, diseño e implementación seguidos por la comunidad OSS también han sido estudiados por Castro y Acuña [2]. Estos autores observan, por ejemplo, que hay ciertos factores comunes en todos los proyectos OSS: (1) la modularidad con la que se diseñan e implementan los sistemas software; (2) la implementación como prioridad en la comunidad OSS; y, (3) los desarrolladores eligen lo que desean diseñar según sus preferencias.

Realmente, ¿puede afirmarse que los procesos OSS difieren del modelo tradicional? Este trabajo apunta en la dirección de responder a esta inquietud, centrándonos en el proceso de mantenimiento, ya que los esfuerzos de la comunidad OSS apuntan principalmente en esa dirección [1]. Además, este trabajo es el segundo de una serie de trabajos donde estudiamos el proceso de desarrollo OSS. Esta serie de trabajos está basada en un único SMS. En el primero de estos [1], abordamos el estudio de los procesos de requisitos, diseño e implementación seguidos por la comunidad OSS. Nuestro objetivo es determinar las diferencias y similitudes existentes entre el proceso de mantenimiento seguido por la comunidad OSS y el establecido por el estándar IEEE 1074:2006 [6]. Este estándar es nuestro modelo de proceso de referencia, pues su influencia se extiende a muchos procesos de desarrollo y es lo que consideramos en este trabajo de investigación como desarrollo tradicional.

Para lograr este objetivo, primeramente necesitamos conocer las actividades que conforman el proceso de mantenimiento seguido por la comunidad OSS, para lo cual hemos realizado un *Systematic Mapping Study* (SMS), siguiendo el procedimiento propuesto por Kitchenham y colegas [7]. Un SMS es una metodología que consiste en investigar la literatura sobre un área de interés particular, con el objetivo de determinar la naturaleza, el alcance y la cantidad de estudios primarios publicados [10]. Hemos elegido esta metodología, porque nos permite identificar y categorizar la literatura mostrando una visión sintetizada del área de investigación que está siendo considerada, con el fin de dirigir la investigación futura. Posteriormente, hemos comparado estas actividades con las establecidas por el estándar IEEE 1074 [6]. Finalmente, he-

mos reconciliado las diferencias existentes y obtenido las características principales que caracterizan al proceso de mantenimiento OSS.

Este trabajo se organiza de la siguiente manera. En la sección 2 se describe el método de investigación. La sección 3 describe el emparejamiento realizado entre las actividades del proceso de mantenimiento del Estándar IEEE 1074 con las actividades análogas del proceso seguido por la comunidad OSS. En la sección 4 se realiza una comparativa entre estas actividades. La sección 5 presenta la discusión del proceso de mantenimiento seguido por la comunidad OSS. En la sección 6 se describen las amenazas a la validez de esta investigación, y finalmente en la sección 7 se presentan las conclusiones.

2 Método de Investigación

El método de investigación aplicado fue el SMS [10]. Este SMS tenía por objetivo responder a la siguiente pregunta de investigación:

RQ: ¿Qué actividades conforman los modelos de proceso OSS?

El proceso de SMS se inició con la identificación de las palabras clave y las cadenas de búsqueda las cuales se derivaron a partir de la pregunta de investigación planteada. Se realizó una búsqueda preliminar, a partir de la cual se obtuvieron algunos artículos que fueron estudiados para determinar los términos más apropiados para el SMS. Estos términos fueron validados y completados por dos expertos investigadores en el área de ingeniería del software. Las cadenas de búsqueda finalmente empleadas fueron:

- *Open Source* AND *Software Process Model*
- *Open Source* AND *Software Development Process*
- *Open Source* AND *Development Process*
- *Free Source* AND *Software Process Model*
- *Free Source* AND *Software Development Process*
- *Free Source* AND *Development Process*

Las bases de datos (BBDD) electrónicas usadas en el SMS fueron: *IEEE Xplore, ACM Digital Library, SpringerLink, Science Direct* y *Scopus*. Cada una de las seis cadenas de búsqueda definidas fue aplicada a cada una de las BBDD seleccionadas. La Tabla 1 presenta para cada una de estas BBDD electrónicas los campos utilizados en la búsqueda. Los campos utilizados no fueron siempre los mismos, ya que dependían de las opciones que brindaba cada una de las BBDD. Los campos *"Abstract"* y *"Title"* fueron utilizados en todas las BBDD.

Durante la búsqueda se fijó como fecha límite de publicación de los artículos el 31 de marzo de 2010. Para la determinación de los estudios primarios que son relevantes a nuestra pregunta de investigación, se utilizaron los siguientes criterios de inclusión y exclusión que se muestran en la Tabla 2.

Tabla 1. Campos de Búsqueda Empleados en cada BBDD

BBDD	Sitio Web	Campos de Búsqueda Seleccionados
IEEE Xplore	http://ieeexplore.ieee.org/	"Index Terms"
ACM	http://portal.acm.org/	"Abstract"
SpringerLink	http://www.springerlink.com/	"All Text OR Abstract"
Science Direct	http://www.sciencedirect.com/	"Abstract OR Title OR Keywords"
Scopus	http://www.scopus.com/	"Article Title OR Abstract OR Keywords"

Tabla 2. Criterios de Inclusión y Exclusión

Criterios de Inclusión
(El título del artículo debe contener las palabras '*open source*' o '*free source*' **OR** Las palabras clave hacían alusión al proceso de desarrollo OSS **OR** El resumen mencionaba alguna cuestión sobre el proceso de desarrollo OSS) **AND** (El artículo describe el proceso de desarrollo OSS **OR** El artículo presenta las actividades del proceso de desarrollo OSS **OR** El artículo presenta las actividades del proceso de desarrollo de software *free source* **OR** El artículo describe el proceso de desarrollo seguido en un proyecto particular OSS **OR** El artículo presenta una propuesta de un proceso de desarrollo OSS)
Criterios de Exclusión
(El artículo no describe el proceso de desarrollo OSS **OR** El artículo no presenta las actividades del proceso de desarrollo OSS **OR** El artículo no presenta las actividades del proceso de desarrollo de software *free source*)

La Tabla 3 muestra para cada BBDD el número de artículos obtenidos al aplicar las seis cadenas de búsqueda, así como el número de artículos candidatos seleccionados. Los artículos candidatos son todos aquellos estudios que cumplen con los criterios de inclusión pero aplicados únicamente sobre el título y las palabras clave. Esta estrategia nos ha permitido filtrar rápidamente el resultado de las búsquedas, al reducir de 12.267 a 619 el número de artículos que evaluar en detalle (solo un 5% del total). Este conjunto de candidatos no contiene duplicados.

Tabla 3. Número Total de Artículos Obtenidos en cada BBDD

Términos de Búsqueda	Encontrados	Candidatos	Estudios Primarios
IEEE Xplore	387	89	7
ACM Digital Library	6.118	282	8
SpringerLink	2.459	147	3
Science Direct	199	21	0
Scopus	3.104	80	1
TOTAL	**12.267**	**619**	**19**

A continuación, se obtuvieron 19 estudios primarios después de aplicar los criterios de inclusión y exclusión de forma rigurosa. Posteriormente, se revisaron las referencias de estos estudios primarios y se encontraron 8 nuevos estudios que no se encontraban en ninguna de las cinco BBDD utilizadas. Así, el número total de estudios primarios ascendió a 27. De estos 27 estudios primarios, se eliminaron 5 debido a que los autores no definían las actividades del proceso OSS, lo que impedía que no se pudiera realizar el análisis de las mismas. Finalmente, el número de estudios primarios utilizados fue de 22, los cuales se muestran en el Apéndice A. Se ha asignado un código a cada uno de dichos estudios primarios para facilitar su referencia en el presente trabajo.

3 Emparejamiento entre las Actividades del Estándar IEEE 1074 con las Actividades del Proceso OSS

De las actividades que conforman los procesos OSS nos hemos centrado en el análisis de las actividades relacionadas con el proceso de mantenimiento. Por una parte, el total de estudios primarios, más de la mitad (55%) estudian proyectos OSS particulares. De este 55% de estudios, más del 80% (10 de 12) cuenta con actividades relacionadas con el mantenimiento. Por otra parte, el 73% del total de estudios primarios encontrados (16 de 22), cuenta con actividades relacionadas con el mantenimiento. La revisión de dichos estudios nos permitió obtener el nombre (cuando estaba disponible) y la descripción de la actividad dada por cada autor. Según la descripción de cada actividad, realizamos un emparejamiento con la actividad análoga del estándar IEEE 1074 [6], teniendo en cuenta para ello no el nombre, sino el objetivo de la actividad de proceso OSS.

Al terminar el emparejamiento, obtuvimos una tabla donde cada una de las actividades del proceso OSS relacionadas con el proceso de mantenimiento fueron emparejadas con una actividad del estándar IEEE 1074. A esta tabla la denominamos *Tabla de Instanciación,* y contiene para cada actividad el nombre y la descripción dado por el autor, así como un comentario. La Tabla 4 ilustra solo un fragmento de esta tabla de instanciación, que por restricciones de espacio no podemos presentar de manera completa.

El emparejamiento supuso un gran esfuerzo, pues algunos autores no dividían la descripción del proceso de desarrollo OSS en actividades, sino que la presentaban como una narrativa, la cual tuvo que ser separada en fragmentos para facilitar su análisis y posterior emparejamiento. En otros casos, cuando los autores si dividían el proceso en actividades, los nombres de éstas no eran los apropiados para las definiciones que los mismos autores aportaban. Por esta razón, se decidió realizar el emparejamiento basándose únicamente en la descripción pero no en el nombre dado por el autor. Adicionalmente, algunas definiciones enmarcadas bajo un solo nombre de actividad debían ser procesadas por partes ya que el autor estaba definiendo realmente varias actividades. De estas dificultades encontradas se han definido las siguientes tipologías:

- CN-CC: Correspondencia **C**orrecta en **N**ombre, **C**orrecta en **C**ontenido.
- IN-CC: Correspondencia **I**nadecuada en **N**ombre, **C**orrecta en **C**ontenido.
- SN-CC: **S**in **N**ombre, **C**orrecta en **C**ontenido.

Tabla 4. Fragmento de la Tabla de Instanciación

Nombre Activ. Estándar IEEE 1074 [6]	Nombre Actividad para el Autor	Definición del Autor	Comentario
Implement Problem Reporting Method	1. Communication and documentation [Mart07]	Another powerful tool in the software process is the bug or issue tracker. An issue tracker lets developers and users report software issues and provides a place to store feature requests so they're not forgotten. We chose the Web-based phpBugTracker system because it's easy to install, has a clean user interface, and can easily handle multiple projects. As developers and users create issues in the system, the issues are given a unique ID and can be assigned a severity and priority. The issue tracker's administrator can give developers extra privileges, so that they can assign status and resolutions to issues. The system also lets you attach files to issues, providing a way to give reproducible test cases and even source patches.	También está en: - Perform Architectural Design - Report Evaluation Results - Reapply SPLCP
⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮
Reapply SPLCP	1. Communication and documentation [Mart07]	Another powerful tool in the software process is the bug or issue tracker. An issue tracker lets developers and users report software issues and provides a place to store feature requests so they're not forgotten. We chose the Web-based phpBugTracker system because it's easy to install, has a clean user interface, and can easily handle multiple projects. As developers and users create issues in the system, the issues are given a unique ID and can be assigned a severity and priority. The issue tracker's administrator can give developers extra privileges, so that they can assign status and resolutions to issues. The system also lets you attach files to issues, providing a way to give reproducible test cases and even source patches.	También está en: - Perform Architectural Design - Report Evaluation Results - Implement Problem Reporting Method
⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮

Partiendo de la Tabla de Instanciación anterior, se procedió a realizar un análisis para determinar si el nombre y la descripción dada por cada uno de los autores a las actividades fue el adecuado. Como resultado de este proceso obtenemos una nueva tabla que denominamos Tabla de Análisis, de la cual se muestra un fragmento en la Tabla 5. Hemos usado las mismas actividades que en la Tabla 4 para ilustrar cómo evoluciona la información en cada una de las tablas y el proceso seguido. Como se puede apreciar, algunos de los nombres de las actividades dados por los autores han sido resaltados en gris, pues el nombre no es el adecuado según la finalidad de la actividad y el emparejamiento realizado. También fragmentos de la definición de ciertas actividades han sido resaltados en gris porque no aportan nada a la definición de la actividad o porque sencillamente definen otra actividad; es decir, el autor ha mezclado actividades con un mismo nombre. Hemos empleado texto resaltado en gris porque en blanco y negro no es posible utilizar códigos de colores.

A continuación, la Tabla 6 presenta el emparejamiento de todas las actividades identificadas del proceso de mantenimiento OSS con las actividades del proceso de mantenimiento del estándar IEEE 1074. Cada una de estas actividades tiene asignada su tipología correspondiente según lo explicado anteriormente, y el código asignado al estudio primario donde se encuentra esta actividad. Por restricciones de espacio no presentamos en la Tabla 6 las definiciones dadas por cada autor a cada actividad.

Tabla 5. Fragmento de la Tabla de Análisis

Nombre Activ. Estándar IEEE 1074 [6]	Nombre Actividad para el Autor	Definición del Autor	Tipología
Implement Problem Reporting Method	1. Communication and documentation [Mart07]	Another powerful tool in the software process is the bug or issue tracker. An issue tracker lets developers and users report software issues and provides a place to store feature requests so they're not forgotten. We chose the Web-based phpBugTracker system because it's easy to install, has a clean user interface, and can easily handle multiple projects. As developers and users create issues in the system, the issues are given a unique ID and can be assigned a severity and priority. The issue tracker's administrator can give developers extra privileges, so that they can assign status and resolutions to issues. The system also lets you attach files to issues, providing a way to give reproducible test cases and even source patches.	IN-CC (Se corresponde con más de una actividad)
⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮
Reapply SPLCP	1. Communication and documentation [Mart07]	Another powerful tool in the software process is the bug or issue tracker. An issue tracker lets developers and users report software issues and provides a place to store feature requests so they're not forgotten. We chose the Web-based phpBugTracker system because it's easy to install, has a clean user interface, and can easily handle multiple projects. As developers and users create issues in the system, the issues are given a unique ID and can be assigned a severity and priority. The issue tracker's administrator can give developers extra privileges, so that they can assign status and resolutions to issues. The system also lets you attach files to issues, providing a way to give reproducible test cases and even source patches.	IN-CC (Se corresponde con más de una actividad)
⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮

Tabla 6. Emparejamiento entre las Actividades del Estándar IEEE 1074 y las Actividades del Proceso de Mantenimiento OSS

Activ. Estándar IEEE 1074 [6]	Nombre Actividad para el Autor	Tipología	Código
Identify Software Improvement Needs	Opening up requirements	IN-CC	[Seny04]
	Consult newsgroups	IN-CC	[Mock02]
	Quality	IN-CC	[Erdo09]
	Identification of work to be done	CN-CC	[Dinh05]
	Phase III: Establishing open source development project	IN-CC	[Gurb06]
Implement Problem Reporting Method	Treating your users as co-developers is your least-hassle route to rapid code improvement and effective debugging	IN-CC	[Raym99]
	Discovering that a problem exists	SN-CC	[Mock00]
	Parallel debugging	IN-CC	[Jorg01]
	Users regarding feature requests, bugs	CN-CC	[Wynn04]
	Change requests	CN-CC	[Mock02]
	Reporting problems or requesting enhancements	CN-CC	[Mock02]
	Consult newsgroups	IN-CC	[Mock02]
	Identifying work to be done	IN-CC	[Mock02]
	Tiered participation	IN-CC	[Simm03]
	Volunteer	IN-CC	[John01]

Tabla 6 (Continuación). Emparejamiento entre las Actividades del Estándar IEEE 1074 y las Actividades del Proceso de Mantenimiento OSS

Activ. Estándar IEEE 1074 [6]	Nombre Actividad para el Autor	Tipología	Código
Implement Problem Reporting Method (Continuación)	Maintenance as evolutionary redevelopment, reinvention and revitalization	CN-CC	[Scac04]
	Quality	IN-CC	[Erdo09]
	Communication and documentation	IN-CC	[Mart07]
	Identification of work to be done	IN-CC	[Dinh05]
	User-initiated change request	CN-CC	[Gurb06]
	Production of packages	IN-CC	[Mong04]
	Bug report	SN-CC	[Yama00]
	Fixing bugs or implementing the proposals	CN-CC	[Yama00]
	Fixing bugs	CN-CC	[Egan04]
	Implement new features	IN-CC	[Egan04]
Reapply SPLCP	Discovering that a problem exists	SN-CC	[Mock00]
	Reporting problems or requesting enhancements	CN-CC	[Mock02]
	Communication and documentation	IN-CC	[Mart07]

La Tabla 7 presenta un resumen de la tipología de las actividades OSS relacionadas con cada una de las actividades del proceso de mantenimiento del estándar IEEE 1074. Este resumen incluye por cada actividad del proceso de mantenimiento tradicional y por cada tipología, el número total de actividades OSS y el número total de estudios primarios diferentes con respecto a esta tipología.

Tabla 7. Resumen de la Frecuencia de las Actividades OSS por Tipología

Nombre de la Actividad en IEEE	Tipología	Nro. Total de Actividades OSS	Nro. Total de Estudios Primarios Diferentes
Identify Software Improvement Needs	CN-CC	1	1
	IN-CC	4	4
Implement Problem Reporting Method	CN-CC	7	6
	IN-CC	11	10
	SN-CC	2	2
Reapply SPLCP	CN-CC	1	1
	IN-CC	1	1
	SN-CC	1	1

Para la actividad *Identify Software Improvement Needs*, el 80% (4 de 5) del total de estudios primarios emparejados en esta actividad nombra erróneamente las actividades. En cuanto a la actividad *Implement Problem Reporting Method*: el 56% (10 de 18) del total de estudios primarios emparejados en esta actividad nombra erróneamente las actividades, mientras que el 33% (6 de 18) cuenta con nombres adecuados se-

gún la descripción dada por el autor. Finalmente, para la actividad *Reapply SPLCP* existe un estudio primario para cada una de las tipologías: CN-CC, IN-CC y SN-CC.

Finalmente, hemos comparado la definición dada por el estándar IEEE 1074 con la definición dada por cada autor según el emparejamiento realizado. De esta comparativa hemos obtenido las diferencias y similitudes entre los procesos de mantenimiento OSS y tradicional, las cuales se describen en el apartado siguiente.

4 Actividades Proceso de Mantenimiento del Estándar IEEE 1074 vs. Actividades Proceso de Mantenimiento OSS

La Tabla 8 ilustra un fragmento de las diferencias y similitudes encontradas entre las actividades del mantenimiento, que por razones de espacio no podemos presentar de manera completa.

Tabla 8. Fragmento de la Comparativa del Estándar IEEE 1074 y las Actividades del Proceso de Mantenimiento OSS

Nombre Activ. Estándar IEEE 1074 [6]	Nombre Actividad para el Autor	Actividad Proceso Tradicional y Definición Dada por el Autor	
		Diferencias	Similitudes
Implement Problem Reporting Method	1. Communication and documentation [Mart07]	En la comunidad OSS los reportes de problemas o solicitudes de mejoras pueden ser realizados por cualquier persona, incluidos los usuarios. Estos diagnostican problemas y a la vez sugieren correcciones, en forma de ideas o de código fuente. En el desarrollo tradicional los usuarios solo reportan los errores, no los corrigen.	El objetivo es el mismo: reportar los problemas encontrados.
	⋮	⋮	⋮
⋮	⋮	⋮	⋮
Reapply SPLCP	1. Communication and documentation [Mart07]	En la comunidad OSS los desarrolladores o los usuarios pueden modificar el estado de un problema (si esta resuelto o no). En el desarrollo tradicional esto no es así.	Se realiza un seguimiento de los problemas reportados. Existe un responsable de esta actividad de seguimiento.
	⋮	⋮	⋮
⋮	⋮	⋮	⋮

En la comunidad OSS cualquier persona, en cualquier momento, puede sugerir o aportar mejoras al sistema software [Gurb06][Mock02][Seny04]. Estas mejoras son comunicadas generalmente a través de listas de correos. Los usuarios finales OSS que actúan como desarrolladores o como los encargados de realizar el mantenimiento, producen continuamente estas mejoras. Las modificaciones o actualizaciones de los sistemas OSS, se convierten luego de revisiones en versiones que son recombinadas con otras para liberar una versión estable. Estas mejoras articulan y adaptan un sistema OSS a las necesidades de los usuarios-desarrolladores, mientras lo reinventan [Scac04]. En algunos proyectos OSS, son los miembros del equipo núcleo los que deciden si adicionan o no una nueva funcionalidad [Dinh05]. En el desarrollo tradicional, la identificación de estas mejoras se realiza con base en diferentes documentos, resultado de otras actividades relacionadas con la planificación del proyecto. Además, son identificadas las herramientas, técnicas y métodos para la aplicación de estas recomendaciones. En algunos proyectos OSS, gracias a sistemas de puntuación, los usuarios saben lo que obtendrán de cada mejora del software. La comunidad OSS

evalúa cada funcionalidad continuamente, y las puntuaciones cambian con el tiempo. De esta manera la calidad es visible [Erdo09]. El estándar IEEE 1074 establece que las recomendaciones de mejora incluyan su impacto en la calidad del software.

En la comunidad OSS, los reportes de problemas o solicitudes de mejoras pueden ser realizados por cualquier persona, incluidos los usuarios, quienes diagnostican problemas y a la vez sugieren correcciones, en forma de ideas o de código fuente [Egan04][John01][Mock02][Raym99][Simm03][Yama00]. Estos reportes de problemas o solicitudes de mejoras se realizan a través de diferentes medios, como listas de correo y grupos de noticias [Mock00]. No hay un repositorio central. Las solicitudes que se encuentran en las listas de correo tienen la mayor prioridad [Mock02]. En el desarrollo tradicional, los usuarios solo reportan errores, no los corrigen. Estos reportes tienen una estructura definida. No todas las personas involucradas en el desarrollo tienen acceso al reporte de mejoras. Los desarrolladores o los usuarios en la comunidad OSS pueden categorizar los problemas y las solicitudes de mejoras, así como también modificar el estado de un problema o un error [Mart07]. En el desarrollo tradicional los usuarios no realizan esta labor.

5 Discusión del Proceso de Mantenimiento OSS

El mantenimiento de software es un proceso generalizado y recurrente en las comunidades de desarrollo OSS como lo es en el desarrollo tradicional. Tal vez esto no es de extrañar teniendo en cuenta que el mantenimiento generalmente es visto como la principal actividad asociada a un sistema software en su ciclo de vida. Sin embargo, el proceso de mantenimiento tradicional del software no encaja con lo que ocurre en la comunidad OSS. Las diferencias encontradas entre el proceso de mantenimiento de software en el desarrollo tradicional y el desarrollo OSS son debidas a la propia naturaleza de cada uno de los procesos. En su lugar, puede ser mejor caracterizar la dinámica general de la evolución OSS como reinvención. La reinvención se produce al intercambiar, examinar, modificar y redistribuir los conceptos y técnicas que han aparecido en el desarrollo tradicional, en publicaciones científicas y libros de texto, conferencias y experiencias de los usuarios-desarrolladores al participar en múltiples proyectos OSS. De este modo, la reinvención es una fuente que emerge continuamente de la adaptación, aprendizaje, y mejora de las funcionalidades y calidad del OSS [Scac04]. No hay distinción entre mantenimiento correctivo y evolutivo.

Los sistemas OSS evolucionan a través de mejoras menores o transformaciones que son expresadas, recombinadas, y redistribuidas a través de múltiples liberaciones con ciclos de vida cortos. Los usuarios finales OSS que actúan como desarrolladores (o como los encargados de realizar el mantenimiento) producen continuamente estas transformaciones. Cualquier persona dentro de la comunidad OSS puede reportar errores, o solicitar mejoras al sistema software. Algunos proyectos OSS como Apache cuentan con sus propias herramientas para el reporte de errores o problemas [Mock02]. Las modificaciones o actualizaciones se expresan como versiones *alpha*, beta que son redistribuidas y revisadas. Estas versiones pueden ser recombinadas con otras transformaciones para liberar una nueva versión estable. Las transformaciones

articulan y adaptan un sistema OSS a lo que sus usuarios-desarrolladores desean que haga, mientras reinventan el sistema [Scac04]. En algunos proyectos OSS, uno o dos desarrolladores realizan el seguimiento periódico de las nuevas peticiones, la eliminación de los reportes de problemas equivocados o mal dirigidos, dan respuesta a las solicitudes sencillas y envían los problemas que consideran críticos a la lista de correo de los desarrolladores[Mock00], [Mock02]. Cuando un problema de cualquier fuente ha sido solucionado, es buscado en los reportes de problemas para que sea incluido en el reporte de incidencias resueltas. El proyecto Apache contaba con una herramienta (BUGDB) para tal fin.

6 Amenazas a la Validez

La validez del SMS presentado en este trabajo se ve amenazada por incluir solamente artículos en inglés, pues todos los términos de búsqueda fueron definidos en este idioma. En este caso no existe riesgo de descarte de estudios primarios, al haber sido contrastados los resultados del proceso de selección con un experto en el área.

No podemos garantizar que todos los estudios primarios relevantes fueron seleccionados durante el proceso de búsqueda. Reducimos esta amenaza siguiendo las referencias en los estudios primarios. Dentro de la comunidad OSS hay una tendencia que aboga por licencias para los manuales y documentación que también sean "*open*", como por ejemplo *creative commons* y similares. Las fuentes consultadas no están soportadas mayoritariamente con este tipo de licencias, por lo que es probable que haya estudios fuera de dichas fuentes.

7 Conclusiones y Trabajos Futuros

Este trabajo de investigación reporta un estudio del proceso de mantenimiento seguido por la comunidad OSS, comparándolo con el prescripto por la ingeniería de software tradicional. Para conocer el proceso de mantenimiento OSS se ha realizado un SMS, cuyo objetivo era responder a la pregunta de investigación: ¿Qué actividades conforman los modelos de proceso OSS? Se encontraron un total de 22 estudios primarios, de los cuales el 73% contaba en sus procesos de desarrollo con actividades relacionadas con el mantenimiento. Estos estudios primarios sirven como punto de partida para realizar un análisis posterior de los procesos OSS y proponer un modelo del proceso seguido por esta comunidad.

La comunidad OSS no sigue los modelos y estándares prescriptivos de la ingeniería del software tradicional. El objetivo principal de esta comunidad es el soporte y mantenimiento de las funcionalidades existentes. No hay una distinción entre mantenimiento correctivo o evolutivo. El mantenimiento en la comunidad OSS puede ser definido como reinvención. Artefactos Web como listas de correos y e-mails son fundamentales en esta reinvención. Los proyectos OSS representan un paradigma alternativo al defendido durante mucho tiempo por la ingeniería del software tradicional.

Los proyectos OSS evolucionan a través de mejoras menores donde participan tanto usuarios como desarrolladores. En algunos proyectos OSS hay una prioridad esta-

blecida para la construcción de las nuevas funcionalidades, o para decidir cuales errores deben resolverse primero de acuerdo a una severidad. La corrección de errores es una tarea de introducción apropiada en un proyecto OSS, si se es una persona a quien le gustan los desafíos. La comunidad OSS también cuenta con actividades análogas a las prescriptas por el estándar IEEE 1074 [6], pero desarrolladas de manera diferente, como por ejemplo que los usuarios pueden contribuir con ideas o con código fuente para corregir los problemas reportados, o que cualquiera dentro de la comunidad pueda asignar prioridades a las mejoras del sistema software.

Dentro de nuestros trabajos futuros, está el realizar una investigación empírica, más de campo, en la que se estudien directamente los registros generados por la comunidad OSS, como listas de correos, wikis, etc. Además de estudiar los trabajos publicados en libresoft.es, http://2012.msrconf.org. Luego, definiremos un modelo de proceso de desarrollo OSS. Una vez definido este modelo, identificaremos un conjunto de técnicas de usabilidad apropiadas para su incorporación en las principales actividades de este modelo. Esta incorporación deberá tener en cuenta las características e idiosincrasias propias del desarrollo OSS, así como también la actividad del proceso donde será ubicada. Se pretende, por ejemplo, determinar cuál o cuáles técnicas de usabilidad pueden incorporarse en el grupo de actividades de mantenimiento.

Agradecimientos. Esta investigación ha sido financiada por el Ministerio de Ciencia e Innovación de España con los proyectos titulados "*Tecnologías para la Replicación y Síntesis de Experimentos en IS*" (TIN2011-23216) y "*Go Lite*" (TIN2011-24139).

Referencias

1. Acuña, S.T., Castro, J.W., Dieste, O., Juristo, N.: A Systematic Mapping Study on the Open Source Software Development Process. In: 16th International Conference on Evaluation and Assessment in Software Engineering (EASE'12), pp. 1-5 (2012)
2. Castro, J.W., Acuña, S.T.: Differences between Traditional and Open Source Development Activities. In: 13th International Conference on Product-Focused Software Development and Process Involvement (PROFES'12), Madrid, Spain, pp. 131-144 (2012)
3. Crowston, K., Wei, K., Howison, J., Wiggins, A.: Free/Libre Open-Source Software Development: What We Know and What We Do Not Know. ACM Computing Surveys. 44 (2), Article 7 (2012)
4. Fuggetta, A.: Open Source Software: An Evaluation. Journal of System and Software. 66, 77-90 (2003)
5. Godfrey, M.V., Tu, Q.: Evolution in Open Source Software: A Case Study. In: International Conference Software Maintenance (ICSM'00), pp. 131-142. San José, CA (2000)
6. IEEE Std 1074:2006: IEEE Standard for Developing Software Life Cycle Processes. IEEE Computer Society (2006)
7. Kitchenham, B.A.: Guidelines for performing Systematic Literature Reviews in Software Engineering Version 2.3. Technical Report S.o.C.S.a.M. Software Engineering Group, Keele University and Department of Computer Science University of Durham (2007)
8. Mockus, A., Fielding, R.T., Herbsleb, J.: A Case Study of Open Source Software Development: The Apache Server. In: 22st International Conference on Software Engineering (ICSE'00), pp. 263-272. Limerck, Ireland (2000)

9. Mockus, A., Fielding, R.T., Herbsleb, J.: Two Case Studies of Open Source Software Development: Apache and Mozilla. ACM Transactions on Software Engineering and Methodology. 11(3), 309-346 (2002)
10. Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M.: Systematic Mapping Studies in Software Engineering. In: 12th International Conference on Evaluation and Assessment in Software Engineering (EASE'08), pp. 71--80 (2008)
11. Potdar, V., Chang, E.: Open Source and Closed Source Software Development Methodologies. In: 26th International Conference on Software Engineering, pp. 105-109 (2004)
12. Scacchi, W.: Understanding the Requirements for Developing Open Source Software Systems. IEE Proceedings-Software. 149(1), 24-39 (2002)
13. Scacchi, W.: Free and Open Source Software Development Practices in the Computer Game Community. IEEE Software. 21(1), 59-67 (2004)
14. Scacchi, W.: Socio-Technical Interaction Networks in Free/Open Source Software Development Processes. In: Acuña, S.T., Juristo, N. (eds.) Software Process Modeling, pp. 1-27. Springer, New York (2005)
15. Scacchi, W., Jensen, C., Noll, J., Elliott, M.: Multi-Modal Modeling of Open Source Software Requirements Processes. In: First International Conference on Open Source Systems, pp. 1-8. Genova, Italy (2005)
16. Tian, Y.: Developing an Open Source Software Development Process Model Using Grounded Theory. Universidad of Nebraska – Lincoln, NB, USA, 143 pp. (2006)

Apéndice A: Estudios Primarios

Este apéndice contiene las referencias de los estudios primarios encontrados al realizar el SMS. A cada uno de los estudios primarios se le ha asignado un Nick para facilitar su referencia en el presente trabajo.

[Dinh05]: Dinh-Trong, T., Bieman, J.M.: The FreeBSD Project: A Replication Case Study of Open Source Development. IEEE Transactions on Software Engineering, 31, 481-494 (2005)

[Egan04]: Egan, S.: The Open Source Development Process. In Open Source Messaging Application Development: Building and Extending Gaim. Chapter 2, Apress, pp. 23-36 (2004)

[Erdo09]: Erdogmus, H.: A Process That Is Not. IEEE Software, 26(6), 4-7 (2009)

[Ezea08]: Ezeala, A., Kim, H., Moore, L.A.: Open Source Software Development: Expectations and Experience from a Small Development Project. In: 46th Annual Southeast Regional Conference on ACM-SE'08, pp. 243-246, Auburn, AL, USA (2008)

[Fitz06]: Fitzgerald, B.: The Transformation of Open Source Software. Forthcoming in MIS Quarterly, 30(3), 1-26. Including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics - 3840 LNCS (2006)

[Gurb06]: Gurbani, V.K., Garvert, A., Herbsleb, J.D.: A Case Study of a Corporate Open Source Development Model. In: 28th ACM International Conference on Software Engineering (ICSE'06), pp. 472-481 (2006)

[John01]: Johnson, K.: A Descriptive Process Model for Open-Source Software Development. Master. Thesis in Computer Science. Department of Computer Science. University of Calgary, 156 pp. (Online) http://sern.ucalgary.ca/students/theses/KimJohnson/thesis.htm (2001)

[Jorg01]: Jorgensen, N.: Putting It All in the Trunk: Incremental Software Development in the FreeBSD Open Source Project. Information Systems Journal, vol. 11(4), 321-336 (2001)

[Lell09]: Lelli, F., Jazayeri, M.: Community Support for Software Development in Small Groups: The Initial Steps. In: 2nd International Workshop on Social Software Engineering and Applications (SoSEA'09), pp. 15-22 (2009)

[Mart07]: Martin, K., Hoffman, B.: An Open Source Approach to Developing Software in a Small Organization. IEEE Software, 24, 46-53 (2007)

[Mock00]: Mockus, A., Fielding, R.T., Herbsleb, J.: A Case Study of Open Source Software Development: The Apache Server. In: 22st International Conference on Software Engineering (ICSE'00), pp. 263-272, Limerick, Ireland (2000)

[Mock02]: Mockus, A., Fielding, R.T., Herbsleb, J.: Two Case Studies of Open Source Software Development: Apache and Mozilla. ACM Transactions on Software Engineering and Methodology, 11(3), 309-346 (2002)

[Mong04]: Monga, M.: From Bazaar to Kibbutz: How Freedom Deals with Coherence in the Debian Project. In: 4th Workshop on Open Source Software Engineering-26th International Conference on Software Engineering (ICSE'04), pp. 71-75 (2004)

[Raym99]: Raymond, E.S.: The Cathedral and the Bazaar. In Cathedral and the Bazaar: Musing on Linux and Open Source by an Accidental Revolutionary, O'Really: Sebastopol, CA, pp. 19-64 (1999)

[Reis02]: Reis, C.R., Mattos Fortes, R.P.: An Overview of the Software Engineering Process and Tools in Mozilla Project. In: Workshop on OSS Development, Newcastle Upon Tyne, UK, pp. 162-182. (Online) http://opensource.mit.edu/papers/ reismozilla.pdf (2002)

[Scac04]: Scacchi, W.: Free and Open Source Development Practices in the Game Community. IEEE Software, 21(1), 59-66 (2004)

[Schw03]: Schweik, C.M., Semenov, A.: The Institutional Design of Open Source Programming: Implications for Addressing Complex Public Policy and Management Problems. Revista First Monday, 8(1), Chicago. (Online) http://firstmonday.org/issues/issue8_1/schweik/ index.html (2003)

[Seny04]: Senyard, A., Michlmayr, M.: How to Have a Successful Free Software Project. In: 11th Asia-Pacific Software Engineering Conference (APSEC'04). IEEE Computer Society, pp. 84-91, Bussan, Corea del Sur (2004)

[Simm03]: Simmons, G.L., Dillon, T.: Open Source Development and Agile Methods. In: 7th IASTED International Conference Software Engineering and Applications, pp. 523-527, Marina del Rey, CA, USA (2003)

[Vixi99]: Vixie, P.: Software Engineering. In Open Sources: Voices from the Open Source Revolution, Chapter 6, 1st Edition, de C. DiBona, S. Ockman, and M. Stone, (eds.) O'Reilly Press: Sebastopol, CA., pp. 91-100 (1999)

[Wynn04]: Wynn Jr., D.E.: Organizational Structure of Open Source Projects: A Life Cycle Approach. In: 7th Annual Conference of the Southern Association for Information Systems, pp. 285-290, Georgia (2004)

[Yama00]: Yamauchi, Y., Yokozawa, M., Shinohara, T., Ishida, T.: Collaboration with Lean Media: How Open-Source Software Succeeds. In: ACM Conference on Computer Supported Cooperative Work (CSCW'00), pp. 329-338 (2000)

Refactorización selectiva de Procesos de Negocio

María Fernández-Ropero, Ricardo Pérez-Castillo, Mario Piattini

Instituto de Tecnologías y Sistemas de Información (ITSI), Universidad of Castilla-La Mancha,
Paseo de la Universidad 4 13071, Ciudad Real, España
{MariaS.Fernandez, Ricardo.PdelCastillo, Mario.Piattini}@uclm.es

Resumen. Los modelos de procesos de negocio se han convertido en uno de los activos más importantes para las organizaciones. Las organizaciones intentan disponer de representaciones precisas de sus procesos de negocio, por lo que deben enfrentarse, durante el ciclo de vida de los procesos de negocio, a defectos en la calidad en dichas representaciones como, por ejemplo, la falta de entendibilidad y modificabilidad. Estos defectos se acentúan cuando los modelos de procesos de negocio han sido extraídos mediante ingeniería inversa (por ejemplo desde los sistemas de información que los soportan parcialmente). En este caso, la refactorización puede ser usada para modificar la representación de los procesos de negocio preservando su comportamiento externo. Este trabajo propone una técnica para seleccionar el conjunto de operadores de refactorización más apropiado en cada caso a fin de maximizar la mejora de entendibilidad y modificabilidad de los modelos de procesos de negocio. La técnica considera un conjunto de medidas presentes en la literatura para evaluar entendibilidad y modificabilidad, y define un conjunto de indicadores para dichas medidas para priorizar la aplicación de cada uno de los operadores de refactorización.

Palabras clave: Modelos de Procesos de Negocio; Refactorización; Entendibilidad; Modificabilidad

1 Introducción

Los procesos de negocio describen una secuencia de actividades de negocio coordinadas, así como los roles y recursos involucrados en ellas, que la organización debe llevar a cabo para conseguir sus objetivos de negocio comunes [1]. Los procesos de negocio son actualmente reconocidos como uno de los activos de negocio intangible que más ventaja competitiva puede aportar a las organizaciones, si estas realizan una gestión apropiada de ellos [2], ya que permiten a las organizaciones adaptarse ágilmente a los cambios. Para gestionar adecuadamente los procesos de negocio es necesario representarlos mediante modelos que simplifiquen la realidad atendiendo a notaciones estándares, siendo BPMN (*Business Process Modeling Notation*) [3] la que se está imponiendo en los últimos años al ser una notación entendida tanto por analistas de sistemas como expertos de negocio.

Las empresas cada vez prestan más atención en describir fielmente sus procesos de negocio y con un grado de calidad óptimo. Es decir, representaciones que eviten de-

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 665-678, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

fectos que afecten a la entendibilidad y modificabilidad, entre otras. Tanto entendibilidad como modificabilidad son las características de calidad consideradas más influyentes en la calidad final percibida de los procesos de negocio [4, 5].

Los defectos de calidad que presentan los modelos de procesos de negocio se manifiestan en mayor medida en aquellos que son extraídos semi-automáticamente mediante ingeniería inversa a partir de, por ejemplo, los sistemas de información que los soportan. El aumento del nivel de abstracción común a cualquier técnica de ingeniería inversa implica lamentablemente cierta pérdida de semántica. Por ejemplo, los modelos de procesos de negocio obtenidos pueden contener un gran número de *gateways* anidados que aumentan su complejidad.

Una solución para mejorar la entendibilidad y modificabilidad de ese tipo de modelos de procesos de negocio es la refactorización. Las técnicas y algoritmos de refactorización modifican la estructura interna de los modelos de procesos de negocio sin alterar su semántica y comportamiento externo, de forma que estos modelos sean más entendibles y/o modificables, y por lo tanto, se reduzcan los costes derivados de su gestión y mantenimiento [6].

Uno de los desafíos de la refactorización de modelos de procesos de negocio es la elección del conjunto de operadores de refactorización a aplicar, así como el orden en el que hacerlo, para un determinado modelo de procesos de negocio. De hecho, los operadores de refactorización propuestos en la literatura son aplicados siguiendo dos enfoques: (1) aplicar el conjunto completo de operadores de refactorización disponibles y (2) aplicar un subconjunto de estos operadores de refactorización bajo la decisión de un experto de negocio. No obstante, ambos enfoques no aseguran que se obtenga la mayor ganancia de calidad posible, es decir, el operador de refactorización que es bueno para mejorar la calidad de un cierto modelo de procesos de negocio puede que no sea apropiado para otro modelo. Este artículo aborda este reto mediante la propuesta de una técnica que aplica operadores de refactorización de forma selectiva basándose en mediciones e indicadores de calidad.

La contribución principal de este artículo se estructura en tres líneas bien definidas. Primero, la técnica propone un conjunto de operadores o reglas de refactorización para modelos de procesos de negocio especialmente obtenidos mediante ingeniería inversa. Segundo, se propone un mecanismo de evaluación de las características de calidad de entendibilidad y modificabilidad mediante un conjunto de medidas obtenidas de la literatura [5, 7, 8]. Tercero, la técnica define de forma heurística un conjunto de indicadores para las medidas anteriores, así como un conjunto de reglas basadas en umbrales que indican si un operador de refactorización debe ser aplicado o no en el modelo de proceso de negocio bajo estudio. Adicionalmente, a fin de facilitar la adopción de esta técnica, se ha desarrollado una herramienta que instrumentaliza dicha técnica.

El resto del documento se organiza de la siguiente manera: la sección 2 resume los trabajos relacionados; la sección 3 detalla las medidas utilizadas para evaluar la entendibilidad y modificabilidad; la sección 4 presenta en detalle los operadores de refactorización; la sección 5 define los indicadores basados en las medidas propuestas y el conjunto de reglas para determinar el/los operador/es de refactorización más ade-

cuados; la sección 6 describe brevemente la herramienta de soporte; y por último, la sección 7 expone las conclusiones y trabajo futuro.

2 Trabajos Relacionados

Hoy en día existe un gran interés en solucionar los problemas de calidad que se pueden presentar a la hora de modelar procesos de negocio. Este interés ha ido creciendo en los últimos años y varios autores han investigado el impacto de las características de calidad en los modelos de procesos de negocio. Todos ellos concluyen que es necesario garantizar la entendibilidad y modificabilidad en un modelo de procesos de negocio, ya que éstos son un artefacto clave en el desarrollo de sistemas de información. El principal objetivo en cada uno de estos trabajos es determinar los parámetros que influyen en las características de calidad entendibilidad y modificabilidad como es el caso de los trabajos [4, 5, 8-10]. Estos trabajos proponen varias métricas para medir estas características de calidad como el número de tareas, el número de conectores, etc.

Otros trabajos como [6, 11, 12] proponen mejorar la calidad de los modelos de procesos de negocio mediante la refactorización y, de esta forma, obtener modelos de procesos más entendibles y más fáciles de mantener. Para poder realizar esto los autores de estos trabajos proponen varios escenarios en los que sería necesario aplicar algún operador de refactorización con el fin de mejorar el modelo y qué operador de refactorización en concreto sería necesario aplicar.

Sin embargo, ninguno de los anteriores trabajos propone utilizar las métricas de calidad que evalúan la entendibilidad y modificabilidad para descubrir los escenarios en los que sería necesario aplicar cada uno de los operadores de refactorización propuestos y, por tanto, ser capaz de aplicar el conjunto más adecuado de operadores de refactorización en lugar de aplicar todos ellos de forma indiscriminada.

3 Medidas de Entendibilidad y Modificabilidad

En esta sección se muestran una serie de medidas definidas para medir la entendibilidad y la modificabilidad de un modelo de procesos de negocio, las cuales han sido validadas de forma empírica por *Rolon et al.* en [7, 8] y *Sánchez-González et al*. en [5]. A continuación se muestra su definición así como sus abreviaturas que son utilizadas a lo largo del documento, divididas dependiendo la característica que evalúa.

3.1 Medidas para evaluar la entendibilidad

- **Número total de eventos** (TNE - *Total Number of events*): Esta variable es relativa al número total de eventos en un modelo de procesos de negocio, es decir, a la suma de eventos de inicio, eventos intermedios y eventos finales.
  - **Número total de eventos de inicio** (TNSE – *Total Number of Start Event*): Relativa únicamente a los eventos de inicio.

  - **Número total de eventos intermedios** (TNIE – *Total Number of Intermediate Event*): Relativa únicamente a los eventos intermedios.
  - **Número total de eventos de fin** (TNEE – *Total Number of End Event*): Relativa únicamente a los eventos de fin.
- **Número de objetos de datos de salida de actividades** (NDOOut - *Number of data objects which are outputs of activities*): Es el número de objetos de datos que son salida de actividades, es decir, que son el destino de un flujo de asociación en el que el origen es una actividad.
- **Número de objetos de datos de entrada de actividades** (NDOIn - *Number of data objects which are inputs of activities*): Es el número de objetos de datos que son entrada de actividades, es decir, que son el origen de un flujo de asociación en el que el destino es una actividad.
- **Número de nodos** (NN - *Number of Nodes*): Esta variable es relativa al número de actividades y elementos de flujo en un modelo de procesos de negocio.
- **Grado promedio de puertas de enlace (Gateway)** (AGD - *Average Gateway Degree*): Expresa el promedio del número de arcos de entrada y salida que tiene una puerta de enlace.
- **Profundidad** (Dep - *Depth*): Es el máximo anidamiento de bloques estructurados en un modelo de procesos de negocio.

3.2 Medidas para evaluar la modificabilidad

- **Nivel de conectividad entre actividades** (CLA - *Connectivity level between activities*): Es el cociente entre el número total de actividades y el número de flujos de secuencia entre actividades.
- **Separabilidad** (Sep - *Separability*): Es el cociente entre el número de vértices de corte (como las puertas de enlace o los eventos intermedios) entre el número total de nodos del modelo de procesos de negocio.

3.3 Medidas para evaluar tanto entendibilidad como modificabilidad

- **Número total de puertas de enlace (Gateway)** (TNG - *Total Number of gateways*): Esta variable es relativa al número total de puerta de enlace en un modelo de procesos de negocio, sea cual sea su tipo.
- **Número de flujos de secuencia procedente de un evento** (NSFE - *Number of sequence flows from event*): Esta variable es relativa al número de flujos de secuencia que tiene como origen un evento.
- **Número de flujos de asociación** (NAF - *Number of association flows*): Es el número de flujos de asociación existentes en un modelo de procesos de negocio.
- **Número de flujos de secuencia procedentes de una puerta de enlace** (NSFG - *Number of sequence flows from gateways*): Esta variable es relativa al número de flujos de secuencia que tienen como origen una puerta de enlace.
- **Densidad** (Den - Density): Es el cociente del número total de arcos en un modelo de procesos de negocio entre el número máximo de arcos posibles.

- **Coeficiente de Conectividad** (CC - *Coefficient of Connectivity*): Es el cociente entre el número total de arcos en un modelo de procesos de negocio entre el número total de nodos.
- **Secuencialidad** (Seq - *Sequentiality*): Es el grado en que se construye el modelo mediante secuencias puras de tareas.

4 Operadores de Refactorización

A continuación se muestran los escenarios (ER) más relevantes en los que se hace necesario aplicar un operador de refactorización, también denominados *smells* en varios trabajos, los cuales que han sido recogidos de la literatura en [6, 11, 13, 14] y adaptados al caso concreto de procesos de negocio obtenidos mediante ingeniería inversa. Junto a estos escenarios se muestra su identificador, para posterior identificación, una descripción, una breve discusión sobre el mismo, las medidas utilizadas para su detección de las comentadas en el apartado anterior, el operador de refactorización (OR) que sería necesario aplicar y la mejora que ofrece la aplicación de dicho operador en la calidad. En la Tabla 1 puede verse gráficamente cada uno de estos escenarios y su resultado tras aplicar el operador de refactorización correspondiente.

4.1 ER1: Nombres no definidos en lenguaje natural de actividades o procesos

Descripción: las actividades y procesos deben tener un nombre característico que revele su propósito. Al obtener los procesos de negocio a partir de ingeniería inversa y teniendo en cuenta que el código fuente fue desarrollado siguiendo la nomenclatura oportuna (clases que comienzan en mayúscula, métodos que empiezan en minúscula, no espacios entre nombres, etc.) las actividades y procesos tendrán un nombre correcto pero sin espacios. Será necesario aplicar un algoritmo que separe las palabras y coloque el primer carácter en mayúscula.

Discusión: estudios proponen utilizar una nomenclatura basada en un formato de verbo-objeto. En nuestro caso, ya se cumple dicha nomenclatura y sólo es necesario separar las palabras para adecuarlas al lenguaje natural.

Operador OR1: separa palabras teniendo en cuenta que una letra mayúscula indica el inicio de una nueva palabra y varias letras mayúsculas consecutivas indican una única palabra, normalmente siglas (Véase Tabla 1).

Medida para su detección: no existe ninguna medida para su detección. Se aplica a todas las actividades y procesos.

Mejora de calidad: entendibilidad.

4.2 ER2: Anidamiento innecesario

Descripción: al obtener los procesos de negocio a partir de ingeniería inversa se incrementa la posibilidad de que existan puertas de enlace (gateways) anidados que aumenten la complejidad. En estos casos, se deben sustituir por alternativas equiva-

lentes más fáciles de comprender por los usuarios. De esta forma, se aumenta la entendibilidad del modelo y hace el mantenimiento del mismo menos costoso.

Discusión: varios estudios demuestran que esta complejidad en el modelo provoca una falta de entendibilidad. Además, existen estudios que proponen medidas para comparar la similitud de varias estructuras.

Operador OR2: si existen dos o más gateways anidados del mismo tipo se sustituyen por un único gateway de ese tipo. (Véase Tabla 1).

Medida para su detección: Den, NN, TNG, NSFG, Dep, Sep, AGD.

Mejora de calidad: entendibilidad y modificabilidad.

4.3 ER3. Malas prácticas en el modelado de procesos de negocio

Descripción: al obtener los procesos de negocio a partir de ingeniería inversa existe la posibilidad de que los modelos no sigan las buenas prácticas recomendadas en el modelado de BPMN.

Discusión: en numerosos artículos recomiendan como buena práctica a la hora de modelar el usar siempre gateways divisores y unificadores. La aplicación de estos gateways debe coincidir.

Operador OR3: Se añade un Gateway exclusivo entre estas actividades. (Véase Tabla 1).

Medida para su detección: CLA.

Mejora de calidad: entendibilidad.

4.4 ER4: Anidamiento innecesario 2

Descripción: al obtener los procesos de negocio a partir de ingeniería inversa se incrementa la posibilidad de que existan gateways anidados que aumenten la complejidad. En estos casos, se deben sustituir por alternativas equivalentes más fáciles de comprender por los usuarios. De esta forma, se aumenta la entendibilidad del modelo y hace el mantenimiento del mismo menos costoso.

Discusión: varios estudios demuestran que esta complejidad en el modelo provoca una falta de entendibilidad. Además, existen estudios que proponen medidas para comparar la similitud de varias estructuras.

Operador OR4: si existen dos o más gateways anidados del mismo tipo se sustituyen por un único gateway de ese tipo. (Véase Tabla 1).

Medida para su detección: Den, NN, TNG, Dep, Sep, AGD.

Mejora de calidad: entendibilidad y modificabilidad.

4.5 ER5. Actividades pequeñas consecutivas

Descripción: al obtener los procesos de negocio a partir de ingeniería inversa existe la posibilidad de que se obtengan numerosas actividades pequeñas consecutivas que no representen gran cantidad de lógica de negocio. Si existen dos o más actividades consecutivas que acceden (leen y/o escriben) únicamente a los mismos datos (data object) y son ejecutadas por el mismo rol puede ser síntoma de que pueden agruparse en una sola actividad que represente el comportamiento de las anteriores.

Discusión: en los modelos de procesos de negocio se puede dar el caso de que una actividad pertenezca a otra, es decir, que tengan semántica similar [14] y por tanto se puedan agrupar.

Operador OR5: estas actividades pequeñas pasan a ser una única actividad (simple) con el mismo comportamiento que las anteriores. En esta refactorización se debe dar un nombre a la nueva actividad que sea representativo de las actividades de las que procede. Una opción es ponerle el nombre de la primera y de la última actividad. (Véase Tabla 1).

Medida para su detección: métrica propuesta por *Smirnov* en [14], NAF, NDOOut, NDOIn, Seq.

Mejora de calidad: entendibilidad y modificabilidad.

4.6 ER6. Fragmentos redundantes

Descripción: en un modelo de procesos de negocio pueden existir fragmentos que contengan la misma lógica de control de flujo. Este hecho hace que el mantenimiento sea más costoso ya que un cambio en el modelo debe ser propagado en todas las ocurrencias de ese fragmento de forma manual y esto puede repercutir en errores de concordancia.

Discusión: una de las razones más comunes para esta redundancia en los modelos se debe a la tendencia a utilizar el copy-paste. Esto provoca que un simple cambio en uno de ellos deba ser reeditado manualmente en cada una de sus ocurrencias. En numerosos artículos se recomienda definir una únicamente vez dicho fragmento de forma global para prevenir errores y mejorar la entendibilidad.

Operador OR6: estos fragmentos del modelo que están duplicados pasan a ser una única actividad compleja que es referenciada en el modelo. (Véase Tabla 1).

Medida para su detección: para su consecución se utilizan métricas capaces de determinar la similitud entre dos fragmentos del modelo como la propuesta por *Dijkman* en [6].

Mejora de calidad: entendibilidad y modificabilidad.

4.7 ER7. Eventos de inicio y/o de fin no conectados

Descripción: al obtener los procesos de negocio a partir de ingeniería inversa existe la posibilidad de que los modelos no estén conectados con los eventos de inicio y/o fin.

Discusión: todos los modelos de procesos de negocio deben comenzar con evento de inicio y finalizar con un evento de finalización.

Operador OR7: realiza la conexión del evento de inicio con la primera actividad y la conexión del evento de finalización con la última actividad. (Véase Tabla 1).

Medida para su detección: TNE, NSFE, CC.

Mejora de calidad: entendibilidad.

4.8 ER8. Varios eventos de finalización

Descripción: al obtener los procesos de negocio a partir de ingeniería inversa existe la posibilidad de que exista más de un evento de finalización [13].

Discusión: al existir más de un evento de finalización se disminuye la entendibilidad del modelo. Por ese motivo, es necesario utilizar fragmentos similares que aporten más entendibilidad.

Operador OR8: se agrupan todas las tareas finales en un gateway exclusivo, es decir, que el flujo terminará cuando uno de los posibles caminos active la puerta de enlace. Aplicar después de OR7. (Véase Tabla 1).

Medida para su detección: TNEE.

Mejora de calidad: entendibilidad.

4.9 ER9. Varias actividades consecutivas

Descripción: al obtener los procesos de negocio a partir de ingeniería inversa existe la posibilidad de que se obtengan numerosas actividades consecutivas. Esto se debe a que cada método es representado como una actividad y puede ser que un método invoque a otro método, este a su vez invoque a un tercero y así sucesivamente.

Discusión: una sucesión de actividades que no son relevantes puede provocar falta de entendibilidad en el modelo. Por ese motivo es preferible agrupar las actividades comprendidas entre la segunda y la penúltima en una sola actividad.

Operador OR9: estas actividades pasan a ser una única actividad (simple) con el mismo comportamiento que las anteriores de forma consecutiva. En esta refactorización se debe dar un nombre a la nueva actividad que sea representativo de las actividades de las que procede. Una opción es ponerle el nombre de la segunda y de la penúltima. (Véase Tabla 1).

Medida para su detección: CLA, Seq.

Mejora de calidad: entendibilidad y modificabilidad.

Tabla 1. Escenarios de Refactorización (ER).

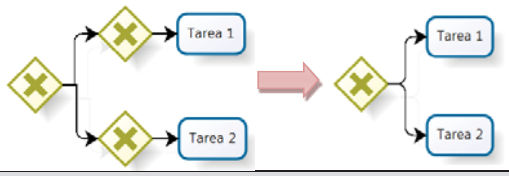
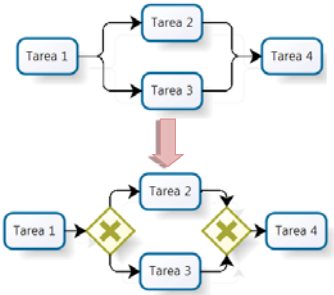
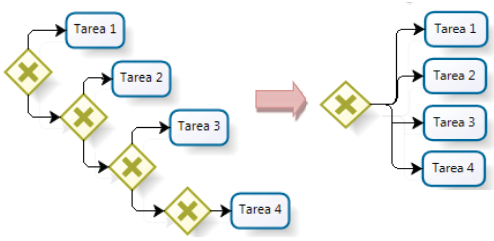
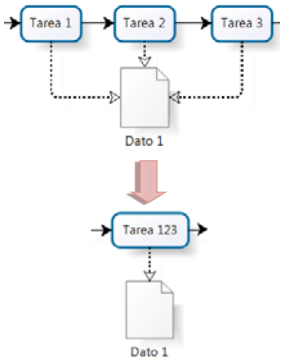
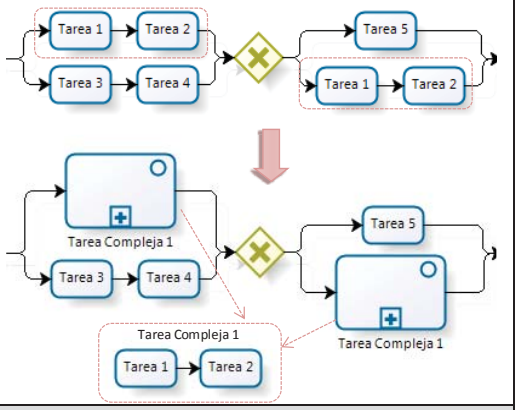
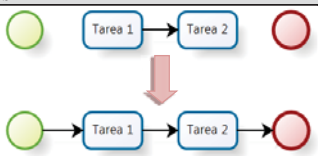
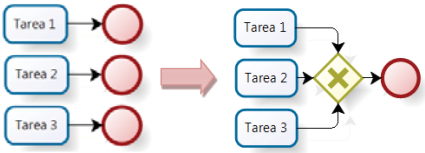
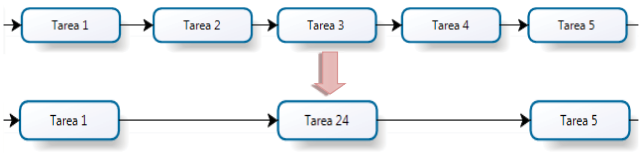
ER1: Nombres no definidos en lenguaje natural de actividades o procesos	**ER2: Anidamiento innecesario**
	
ER3. Malas prácticas en el modelado de procesos de negocio	**ER4: Anidamiento innecesario 2**
	
ER5. Actividades pequeñas consecutivas	**ER6. Fragmentos redundantes**
	
ER7. Eventos de inicio y/o de fin no conectados	**ER8. Varios eventos de finalización**
	
ER9. Varias actividades consecutivas	
	

Tabla 2. Relación entre medidas y operadores de refactorización.

	OR1	OR2	OR3	OR4	OR5	OR6	OR7	OR8	OR9
TNE							♦	♦	
TNG		♦		♦					
NSFE							♦		
NMF					♦				
NSFG		♦							
NDOOut					♦				
NDOIn					♦				
CLA			♦						♦
NN		♦		♦					
Den		♦		♦					
CC							♦		
AGD		♦		♦					
Sep		♦		♦					
Seq					♦				♦
Dep		♦		♦					
Dijkman						♦			

A modo resumen se presenta en la Tabla 2 una matriz de operadores de refactorización y medidas en la que se especifica qué medidas intervienen o afectan en la aplicación de un determinado operador de refactorización.

5 Indicadores y Reglas de Refactorización Selectiva

Una vez descritos los posibles operadores de refactorización junto las medidas para su detección se muestran en la Tabla 3 qué valores han de tener de dichas medidas para activar cada uno de los operadores de refactorización. Además, se muestra la mejora realizada en base a las medidas. En la primera columna aparece el identificador de la refactorización, en la segunda columna aparecen las medidas que son necesarias medir para promover su aplicación, en la tercera columna se muestran los intervalos a los que deben pertenecer los valores de las mediciones para que se realice dicha refactorización, y en la última columna se muestra los nuevos valores de las medidas tras la aplicación de la refactorización en base a los valores iniciales. Estos intervalos han sido establecidos de manera heurística tras la observación de numerosos modelos de procesos de negocio reales. El subíndice "f" simboliza el valor final de la medida y el subíndice "i" simboliza el valor inicial de la medida.

En algunos casos los valores de las mediciones pueden ser insuficientes para aplicar la refactorización por lo que haría falta la opinión de un experto. Estos casos son simbolizados en la Tabla 3 mediante la etiqueta "*Expert*".

Tabla 3. Intervalos para la activación de cada uno de los operadores de refactorización. Se asume un operador lógico OR entre los distintos intervalos para la activación de cada operador.

Operador Refact.	Medida para su detección	Intervalos para su activación	Mejora en las medidas
OR2	Den, NN, TNG, NSFG, Dep, Sep, AGD.	Si NSFG = 1 Si AGD = 2 Si $\frac{TNG}{NN} > \frac{1}{2}$ Si Den = 2 & Dep = 2 → *Expert*	$NN_f < NN_i$ $NSFG_f = 1$ $TNG_f < TNG_i$ $Den_f < Den_i$ $Dep_f < Dep_i$ $AGD_f > AGD_i$
OR3	CLA	Si CLA < 1	$CLA_f > CLA_i$ $TNSF_f > TNSF_f$ $TNG_f > TNG_i$
OR4	Den, NN, TNG, Dep, Sep, AGD.	Si $\frac{TNG}{NN} > \frac{1}{2}$ Si AGD = 3 → *Expert* Si Den = 2 & Dep = 2 → *Expert*	$NN_f < NN_i$ $NSFG_f > NSFG_i$ $TNG_f < TNG_i$ $Den_f < Den_i$ $Dep_f < Dep_i$ $AGD_f > AGD_i$
OR5	NAF, NDOOut, NDOIn, Seq.	Si NAF > NDOOut + NDOIn & Seq ≥ 3	$NAF_f < NAF_i$ $NN_f < NN_i$
OR6	Medida propuesta por *Dijkman* [6]	Reglas propuestas por *Dijkman* [6]	$NN_f < NN_i$ $Den_f < Den_i$
OR7	TNE, NSFE, CC	Si TNE > NSFE Si CC < 1	$NSFE_f > NSFE_i$ $CC_f > CC_i$
OR8	TNEE	Si TNEE > 1	$TNEE_f = 1$ NNf <NNi $CC_f > CC_i$
OR9	CLA, Seq	Si CLA = 1 & Seq ≥ 5	$Seq_f < Seq_i$ $CLA_f = CLA_i$ $NN_f < NN_i$

6 Herramienta de Soporte

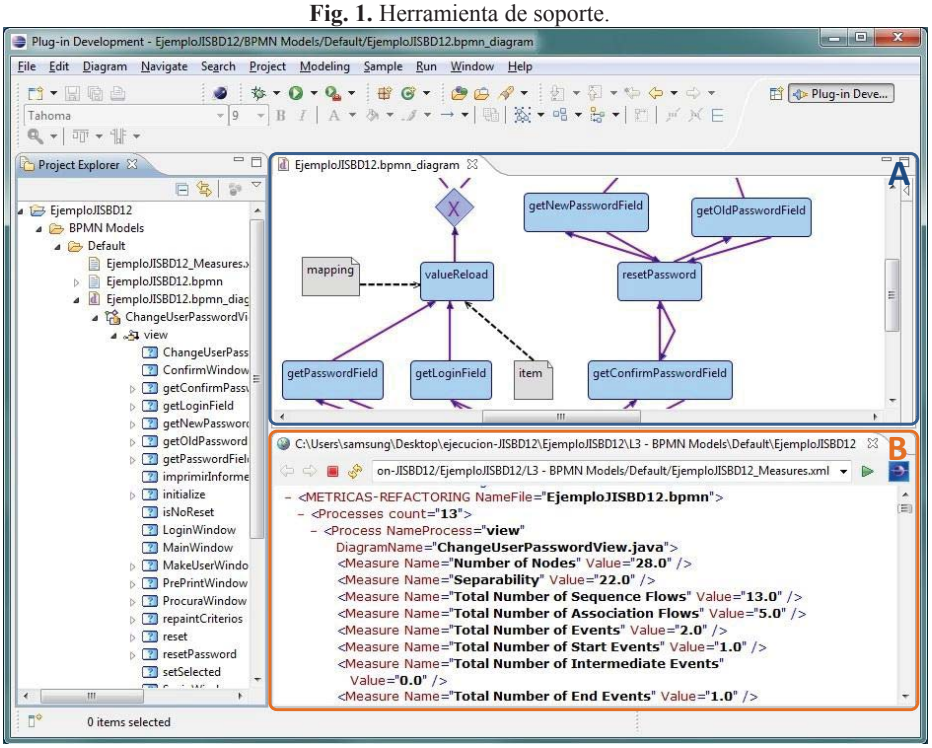
Actualmente se está trabajando en la implementación de una herramienta que permita calcular las medidas comentadas en la sección 2 respectivas a un modelo de proceso de negocio concreto. Junto a estas métricas también se están implementando los operadores de refactorización mostrados en la Tabla 1 y el conjunto de reglas mostradas en la Tabla 3 para su activación. Esta herramienta está siendo implementada como un plug-in para Eclipse™ de tal forma que sea compatible con otras herramientas de ingeniería inversa que han sido implementadas como plug-in para este entorno de desarrollo y obtienen procesos de negocio a partir de código fuente (por ejemplo, la herramienta MARBLE [15]).

En la Fig. 1 se muestra un extracto de la herramienta desarrollada. En la parte A de la figura se muestra el modelo de procesos de negocio que se pretende refactorizar

representado gráficamente bajo la notación BPMN. En la parte B de la figura se encuentran los valores de las medidas implementadas correspondientes a dicho modelo de procesos de negocio mostrado como ejemplo.

El objetivo de esta herramienta es realizar casos de estudios que permitan refinar y/o modificar las reglas de activación definidas en este trabajo y obtener los indicadores más adecuados para cada una de las medidas.

Fig. 1. Herramienta de soporte.



7 Conclusiones y Trabajo Futuro

El trabajo presenta una técnica de refactorización selectiva de modelos de procesos de negocio basada en mediciones de calidad. Esta idea se aparta de la idea convencional de aplicar operadores de refactorización de forma indiscriminada o mediante un experto de negocio. La idea es guiar las refactorizaciones usando los valores de las mediciones de las características de entendibilidad y modificabilidad con el fin de asegurar un mejor resultado final.

La técnica propone utilizar un conjunto de 18 medidas para evaluar las características de calidad de entendibilidad y/o modificabilidad de los modelos de procesos de negocio. Posteriormente, la técnica define un conjunto de indicadores y reglas basadas en dichas medidas con el fin de determinar el conjunto más apropiado de operadores de refactorización que permita aumentar en mayor medida el grado de calidad de un

determinado modelo de proceso de negocio. Mientras que las medidas para estas características de calidad se han extraído mediante una revisión sistemática de la literatura, los umbrales de los indicadores obtenidos a partir de esas medidas así como las reglas para aplicar unos operadores de refactorización han sido definidos heurísticamente.

Este trabajo piloto, en el cual se ha desarrollado además una herramienta de soporte, está permitiendo en la actualidad aplicar la técnica de refactorización selectiva a diferentes modelos de proceso de negocio obtenidos mediante ingeniería inversa desde sistemas de información reales. Esta validación empírica en curso permitirá refinar y ajustar los indicadores basándonos en evidencias reales.

Además, como otras líneas de trabajo futuro se ha planificado: (i) determinar la mejora obtenida con cada uno de los operadores de refactorización con el fin de establecer no sólo el conjunto que sería necesario aplicar, sino también el orden en el que aplicarlos para obtener mayor grado de calidad; y (ii) incorporar nuevas medidas relativas a otras características de calidad como la testabilidad, perteneciente a la mantenibilidad, o la seguridad.

Agradecimientos

Este trabajo ha sido soportado por el programa FPU así como los siguientes proyectos I+D: ALTAMIRA (JCCM, PII2I09-0106-2463), MOTERO (JCCM and FEDER, PEII11-0366-9449), PEGASO/MAGO (TIN2009-13718-C02-01) y MOTERO (JCCM and FEDER, PEII11-0366-9449).

Referencias

1. Weske, M., *Business Process Management: Concepts, Languages, Architectures*. 2007, Leipzig, Alemania: Springer-Verlag Berlin Heidelberg. 368.
2. Jeston, J., J. Nelis, and T. Davenport, *Business Process Management: Practical Guidelines to Successful Implementations*. 2nd ed. 2008, NV, USA: Butterworth-Heinemann (Elsevier Ltd.). 469.
3. OMG. *Business Process Modeling Notation Specification 1.0*. 2006; Available from: http://www.omg.org/bpmn/Documents/OMG_Final_Adopted_BPMN_1-0_Spec_06-02-01.pdf.
4. Reijers, H.A. and J. Mendling, *A study into the factors that influence the understandability of business process models.* Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, 2011(99): p. 1-14.
5. Sánchez-González, L., et al., *Quality assessment of business process models based on thresholds.* On the Move to Meaningful Internet Systems: OTM 2010, 2010: p. 78-95.
6. Dijkman, R., et al., *Identifying refactoring opportunities in process model repositories.* Information and Software Technology, 2011.
7. Rolon, E., et al. *Evaluation measures for business process models*. 2006: ACM.
8. Rolon, E., et al. *Prediction models for BPMN usability and maintainability*. 2009: IEEE.
9. Mendling, J., H. Reijers, and J. Cardoso, *What makes process models understandable?* Business Process Management, 2007: p. 48-63.

10. Mendling, J. and M. Strembeck. *Influence factors of understanding business process models*. 2008: Springer.
11. Weber, B., et al., *Survey paper: Refactoring large process model repositories.* Comput. Ind., 2011. **62**(5): p. 467-486.
12. Leopold, H., S. Smirnov, and J. Mendling, *Refactoring of process model activity labels*, in *Proceedings of the Natural language processing and information systems, and 15th international conference on Applications of natural language to information systems*. 2010, Springer-Verlag: Cardiff, UK. p. 268-276.
13. Koehler, J., et al., *Combining Quality Assurance and Model Transformations in Business-Driven Development*, in *Applications of Graph Transformations with Industrial Relevance*, S. Andy, et al., Editors. 2008, Springer-Verlag. p. 1-16.
14. Smirnov, S., H. Reijers, and M. Weske. *A semantic approach for business process model abstraction*. 2011: Springer.
15. Pérez-Castillo, R., et al., *MARBLE. A Business Process Archeology Tool*, in *27th IEEE International Conference on Software Maintenance (ICSM 2011)*. 2011: Williamsburg, VI. p. 578 - 581

Accessibility and Internationalization in Requirements Engineering Tools

José Luis Fernández-Alemán, Juan M. Carrillo De Gea, Joaquín Nicolás, Ambrosio Toval, Diego Alcón, and Sofia Ouhbi

Faculty of Computer Science, Regional Campus of International Excellence "Campus Mare Nostrum", University of Murcia, Murcia, Spain
aleman@um.es, jmcdg1@um.es, jnr@um.es, atoval@um.es, d.alconcazorla@um.es, sofia.ouhbi@um.es http://www.um.es/giisw/

Abstract. In recent years, there has been a significant increase in software development in collaborative and globally distributed settings. Thus, there is a growing interest on accessibility and internationalization issues in Requirements Engineering (RE) tools. The main contribution of this paper is to shed light on RE tools' accessibility and internationalization, which are key capabilities concerning Global Software Development (GSD). To the best of our knowledge, this is the first manuscript on this subject. A 147-item checklist based principally on the features covered by the standard ISO/IEC 9241-171 was used to assess 13 RE tools. Then, a descriptive statistical study was carried out to provide comparability. Bivariate correlation tests were also applied to measure the association between different variables. A major margin for amelioration was found by current RE tools, mainly with regard to the input, output and internationalization features. The outcome of this research shows that there is a lack of adequacy to the accessibility and internationalization needs. In future work, the scope of the study will be extended to cover other capabilities on RE tools and address the multi-cultural challenges in the GSD.

Keywords: Requirements Engineering Tools, Accessibility, Internationalization

1 Introduction

The Information Technology (IT) and the software industry are now truly global, as it is the Software Engineering (SE). The diversity of cultures and the dispersion in time and space involved in globally distributed software development require new techniques, tools and practices from various disciplines to meet the challenges and opportunities offered by global SE [9]. So, multidisciplinary research is essential to increase the knowledge of cooperative and distributed work and how it can be supported by IT [17].

A significant increase in Global Software Development (GSD) has been observed in recent years. This is a phenomenon that brings significant benefits to

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 679-692, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

organizations, whether in the form of lower personnel costs by wage differentials between countries, greater availability of skilled professionals than whom are accessible only in the immediate environment, better access to important markets or customers because of headquarters localization proximity, longer working timetables when the teams are in locations with different time zones, and eventually enriching diversity of experiences, techniques and skills of the distributed individuals involved (stakeholders). However, working in a global context has advantages but also drawbacks [11]. There are many obstacles to overcome regarding knowledge management, configuration management, cultural differences, communication, and many others. GSD challenges traditional techniques of SE [11], affecting all activities of the software production process, from RE to the delivery of the software. It is therefore necessary to observe all the software processes through the prism of GSD and devise new working methods, or adapt the existing ones conveniently.

With specific regard to the RE in the GSD, Cheng and Atlee [8] identified globalization as one of the hot spots in research in this discipline. For these authors, to overcome the difficulties posed by the GSD in the RE is essential for achieving an efficient distributed development of software products. A number of authors [22] point out that many recent advances in Collaborative Software Engineering (CoSE) have to do with the creation of tools that provide support for certain collaborative practices. Therefore, making an effort to investigate the potential of RE tools to alleviate some of the lacks that have been identified in the field of RE for GSD has been revealed as necessary. Carrillo et al. [13] found deficits in the user interface of the current RE tools.

The main contribution of this paper is to shed light on the RE tools' accessibility and internationalization capabilities for GSD environments. A study of the general RE capabilities of RE tools has already been conducted [13]. However, to the best of our knowledge, this is the first manuscript that presents an assessment of accessibility and internationalization in current RE tools, concluding with a description of its strengths and weaknesses.

The rest of this paper is organized as follows: first, Section 2 defines properly accessibility and internationalization. Section 3 describes the research methodology, including the research goals, the research questions guiding this study, the instruments and the classification framework used, and the experimental procedure. Next, Section 4 presents the results of the study. Then, in Section 5, we discuss the findings and the broader implications of this work in more detail. Finally, Section 6 draws some conclusions and outline future work.

2 Accessibility and internationalization

Accessibility and internationalization are two important capabilities required by tools. According to the ISO TS 16071 [4], accessibility is defined as the usability of a product, service, environment or facility by people with the widest range of capabilities. Usually, the term "Design for All" is used to address the goal of enabling maximum access to the maximum number and diversity of users,

irrespective of their skill level, language, culture, environment or disability. According to the standard ISO 9241, several approaches can be used to increase the accessibility of a human-system interface, namely: (1) adopting a human centered approach to design (ISO 13407); (2) following a context-based design process; (3) providing the capacity for individualisation (ISO 9241-110); and (4) offering individualised user instruction and training.

A common definition for internationalization (i18n) [16] is: "the process of designing an application so that it can be adapted to various languages and regions without engineering changes". There are no many international organizations working on the standardization of i18n. A few standardization bodies have created specific groups about this topic. The most important sources used to populate the checklist used in this study were:

1. The W3C Internationalization (I18n) Activity [3] works with W3C working groups and liaises with other organizations to facilitate using Web technologies with different languages, scripts, and cultures. Web Content Accessibility Guidelines 2.0 covers also a wide range of recommendations for making Web content more accessible.
2. ISO/IEC 24751:2008 [18] [2] [15] [16]. In ISO/IEC 24751, disability is not considered as a personal trait but as a consequence of the relationship between a learner and a learning environment or resource delivery system. In particular, a lack of language proficiency can produce a mismatch of personal needs and preferences with education digital resources.
3. ISO 9241:2008. Ergonomics of human-system interaction. The Part 151 [12]: Guidance on World Wide Web user interfaces provides guidance on the human-centered design of Web interfaces with the aim of increasing usability. Section 10 (General design aspects) provides guidelines for cultural diversity and multilingual use in Web application.
4. A number of standards of the European Committee for Standardization (CEN) [1] provide recommendations. Some examples are: CWA 14928 - Review on SIF infrastructure, architecture, message processing and transport layer; CWA 14929 - Internationalization of SIF and harmonization with other specifications/standards. Regarding vocabulary: CWA 14590:2002 - Description of language capabilities.
5. A reusable repository (catalog) of software i18n requirements proposed in [24]. This document, named EI-CAT (*E-learning Internationalization CATalog*), has been constructed according to the best requirements engineering practices.
6. ISO 9241-171:2008 [23] provides ergonomics guidance and specifications for the design of accessible software. This standard can be used by those responsible for the specification, design, development and evaluation of software applications. The use of assistive technologies as an integrated component of interactive systems is also addressed.

Goal	The goal is to depict the accessibility and i18n capabilities on RE tools for GSD
Question	Do current RE tools address accessibility and i18n in the GSD?
Metric	The accessibility and i18n capabilities supported by the RE tools
Goal definition template	
Object of study	The objects studied are the RE tools' user interfaces
Purpose	The purpose is to characterize accessibility and i18n, two keys user interface features for RE tools in GSD settings and to evaluate the scene of RE tools
Quality focus	The quality focus is the suitability of the RE tools' user interface for GSD settings
Perspective	The perspective is from the point of view of the practitioners
Context	The study is run using RE tools appearing in at least one on the databases described on Table 2

Table 1. Goal/Question Metric template

3 Research methodology

3.1 Research goals and questions

The research goals were outlined using the Goal/Question Metric (GQM) framework [6]. The GQM template [5] of the study is shown in Table 1. Bearing in mind that providing practitioners with high-quality software products is essential to address the challenges of the GSD and to achieve an efficient distributed development, the potential of the current RE tools to alleviate the lacks in the user interface for GSD is investigated. We will try to answer the following research questions:

- **RQ1**. To which extent RE tools support accessibility features?
- **RQ2**. To which extent RE tools support i18n features?

3.2 Instrumentation

The problem of selecting the set of tools to be included in the study has been faced by searching for and consulting seven well-known databases which contain a series of lists of RE tools: *Ian Alexander*, *Alarcos Research Group*, *INCOSE*, *Ludwig Consulting Services*, *Qaguild*, *Volere*, *@WEBO* and *University of Murcia Software Engineering Research Group*, corresponding to the acronyms A, G, I, L, Q, V, W and M, respectively, in Table 2. At the moment of accessing these databases, from July–August 2010, the number of tools specified in Table 2 was retrieved. A large number of tools appeared in more than one database simultaneously. In addition, a number of the tools listed were not in force, there was no vendor or person responsible for them, or their name had changed. After reviewing the whole set of tools included in the databases depicted in Table 2, a total number of 38 RE tools was chosen for evaluation [14]; in the meantime, their vendors' contact details were collected. Both search and retrieval processes were carried out carefully in order to be accurate and impartial.

In the subsequent stage of the study the analysis method to be applied was selected. DESMET [20] is a method designed by Barbara Kitchenham for evaluating SE methods and tools. In particular, the evaluation type "qualitative

DB	Web	# tools
A	http://easyweb.easynet.co.uk/ iany/other/vendors.htm	67
G	http://sites.google.com/site/toolsgsd/tools-1/software-requirement-tools	7
I	http://www.incose.org/ProductsPubs/products/rmsurvey.aspx	34
L	http:www.jiludwig.comRequirements_Management_Tools.html	40
Q	http://qaguild.com/Toolsdirectory/RequirementManagementTools.htm	7
V	http://www.volere.co.uk/tools.htm	71
W	http://www.atwebo.com/case.htm#Requirements%20Capture	41
M	http://www.um.es/giisw/EN/re-tools-survey/	38

Table 2. Searched databases to identify tools. DB: Database

experiment" is based on the assumption that there are identifiable and measurable properties of the software product or process that are expected to change as a result of using the methods or tools under evaluation (tools in our case). Feature analysis will be referred to as qualitative because it usually requires a subjective assessment of the relative importance of different features and how well a feature is implemented in the corresponding tool.

The ISO 9241 standard, part 151 [12] and part 171 [23], the W3C Internationalization Activity [3], the ISO/IEC 24751 standard [18], a number of CEN standards [1], and the EI-CAT [24] were used to obtain the assessment framework since they provide recommendations to design accessible and international software and is applicable to all software application domains.

3.3 Classification Framework

Recently, a new framework for the evaluation of the RE tool's capabilities has been proposed, the ISO/IEC TR 24766:2009 [19]. According to the ISO/IEC TR 24766, we can classify the RE tool capabilities into six major categories: requirements elicitation, requirements analysis, requirements specification, requirements verification and validation, requirements management, and other tool capabilities. However, this document contains a very limited number of features relating to the accessibility and i18n support provided by the tools.

After searching for a more general classification framework, the ISO/IEC 9241-171 [23] standard of accessibility was found. Starting from the checklist provided by ISO/IEC 9241-171 (parts 8, 9, 10 and 11), a 139-item questionnaire was designed, grouped in four categories (General guidelines, Inputs, Outputs, Documentation):

- *8 General guidelines*: Input/output alternatives, names and labels for user interface elements, user preference settings, special considerations for accessibility adjustments such as sizes and colours, general control and operation guidelines, compatibility with assistive technology, closed systems (systems that does not allow user installation of assistive technology software).
- *9 Inputs*: Alternative input options, keyboard focus, keyboard input, pointing devices.
- *10 Outputs*: General output guidelines, visual output (displays), text/fonts, color, window appearance and behavior, audio output, text equivalents of audio, media and animation, tactile output.

Id.	Description
1	The tool shall design for cultural diversity and multilingual use.
2	The tool shall allow to create a PNP (Personal Needs and Preferences) Statement. PNP description can be created in a [variety of ways].
3	The tool shall allow the individualization and adaption, through use of [profiles] (user profiles or group profiles).
4	The tool environment shall allow to adjust the user interface or configuration (with respect to presentation, control methods, structure, access mode, and learner supports, for example).
5	The tool shall allow to change the [character encoding] (ISO, UTF-8, etc.)
6	The tool shall allow taking into account all information that may vary depending on a different educational/cultural context. These include Address, Name, School Info, Student Section Enrollment, Term Info, etc.
7	The tool shall use appropriate formats, units or measurement or currency for international audience.
8	The tool shall provide help, FAQs in [multiple languages].

Table 3. Internationalization requirements

– *11 Online documentation, help and support services*: Accessible training material, understandable documentation, accessible technical support and client support services which meet applicable accessibility standards.

In addition, a set of i18n capabilities has been defined in order to give support for the process of RE in global environments. These were selected from the main related SE standards provided by international standardization bodies and consortia active, as well as a reusable repository (catalog) of software i18n requirements [24]. Therefore, a set of 8 questions regarding the tools' i18n capabilities has been added to the questionnaire, by transposing the features extracted from ISO/IEC 24751-2, ISO 9241-151, CWA 14928 and other standardization efforts to a list of enquiries. These i18n questions are based on the requirements listed in Table 3.

A total of five categories was established and the number of capabilities falling within each category is shown in Table 4. Therefore, this study used five variables, one for each category evaluated: I (Internationalization), GG (General guidelines), In (Inputs), O (Outputs) and D (Documentation). The questionnaire can be downloaded from the Web site http://www.um.es/giisw/EN/re-tools-survey/AccIntChecklist.xlsx.

Description	# questions
Accessibility	**139**
General guidelines	49
Inputs	38
Outputs	45
Online documentation, help and support services	7
Internationalization	**8**
Total	**147**

Table 4. Structure of the questionnaire

3.4 Research Procedure

The checklist was prepared by the authors between June–July 2011. Also, there was one person in charge of technical-related tasks—tools installation and configuration in a web server. Contact with the tools representatives was always done by e-mail, using the contact information previously collected. The registration procedure was done on June 20th, 2011. A total of 5 tools were web-based applications and 32 desk applications. Installing problems were found in 24 desk tools. The reasons for which these tools could not be installed were: (1) server connection problems, (2) unresolved dependencies, (3) authentication problems in passwords provided by the vendors and (4) elusive problems when running the tool. In conclusion, the original amount of tool candidates for participation (38) was reduced to 13 (Table 5).

No.	Tool	Country	Desk/Web
1	Aligned Elements	Switzerland	Desk
5	Bright Green Projects	UK	Web
21	QFDcapture	USA	Desk
23	RaQuest	Japan	Desk
24	IBM Rational DOORS	USA	Web
25	ReqMan	USA	Web
28	RTIME	USA	Desk
30	RMTrak	USA	Desk
34	TestTrack RM	USA	Desk
35	TopTeam Analyst	USA	Desk
36	TraceCloud	USA	Web
37	TrackStudio	Russia	Desk
38	VisibleThread On-premise/On-demand	USA	Web

Table 5. Tools evaluated

4 Results

A descriptive statistical approach was used to test the research questions of this study. A case study on Web application domain was selected to carry out and conduct the evaluation process. Each one of the 147 questions were answered by the assessors: Yes (1), No (0) or Partly (0.5) by interacting with and navigation through the 13 RE tools finally installed. Table 7 in Appendix A summarizes the result of our research.

As shown in Figure 2, the evaluation of the 13 RE tools, scored by category of features, has been incorporated into the same graph to be compared with each other. For each category of features c, the score of each tool t is calculated using Formula (1):

$$\frac{\sum_{q=1}^{NQ(c)} score(t,q)}{NQ(c)} \tag{1}$$

NQ(c) is the number of questions of the category c, where the expression $score(t,q) \in \{0, 0.5, 1\}$ is the score of the tool t in the question q. This score

	I	GG	In	O	D
I	1,000				
GG	-0,197*	1,000			
In	-0,543*	0,705**	1,000		
I	-0,291*	0,892**	0,838**	1,000	
D	0,185*	0,212*	-0,120*	0,157*	1,000

Table 6. Correlation matrix (N=13). **: Correlation is significant at the 0.01 level (1-tailed). *: Correlation is significant at the 0.05 level (1-tailed)

is discretised on a 3-interval scale using a global unsupervised discretisation method [10], a variation of the equal width interval binning in which the lower and upper bins are shorter than the others with the intention of discriminating extreme scores:

$$discretise(s) = \begin{cases} \text{High}, & s \in (0.7, 1]; \\ \text{Medium}, & s \in (0.3, 0.7]; \\ \text{Low}, & s \in [0, 0.3]. \end{cases}$$

The average score for Internationalization, General guidelines, Inputs, Outputs and Online documentation was 0.245, 0.337, 0.121, 0.111 and 0.280, respectively. The scores obtained show a low i18n and accessibility level concerning the RE tools analyzed, except in General guidelines which achieves a medium score. This result answers **RQ1** and **RQ2**. Notice that scores in all categories are equal or lower than 0.35, except i18n in Rational DOORS, which receives a score of 0.438.

Bivariate correlation tests were also applied to measure the association between the variables identified. Table 6 shows that there is strong positive correlation between Inputs, Outputs and Online documentation. There is not appreciable correlation between the rest of variables.

5 Discussion

Accessibility capabilities are offered in a similar degree by RE tools, i.e. in general, there are no remarkable differences among tools. However, the level of accomplishment of the accesibility recommendations is different depending on the area. Thus, General Guidelines and Documentation are better supported areas than Inputs and Outputs. It could be explained by the fact that the first group of features is easier to implement than the second group of features. General Guidelines include recommendations such as user preference settings (e.g. enable easy individualisation of user preference settings, enable adjustment of sizes and colours of common user interface elements) and general control and operation guidelines (e.g. optimise the number of steps required for any task, provide “undo” and/or “confirm” functionality), which are nowadays commonly accepted and provided by software companies. This same statement can be ap-

plied to Documentation, that includes recommendations on documentation and help and support services.

As shown in Figure 2, Inputs and Outputs seem to be the more suitable areas for RE tools' improvement, in particular, the configuration and integration with external devices. We think that this could be due to the fact that RE tools are traditionally more oriented towards textual, natural language requirements management thus requiring less interaction capabilities with the user. Inputs include guidelines such as keyboard input (e.g. enable full use via keyboard, enable remapping of keyboard functions) and pointing devices (e.g. provide direct control of pointer position from external devices, provide easily-selectable pointing device targets), while Outputs include recommendations such as general output guidelines (e.g. avoid seizure-inducing flash rates, enable user control of time-sensitive presentation of information) and visual output (e.g. enable users to adjust graphic attributes, provide a visual information mode usable by users with low visual acuity).

From Table 7, it can be noticed that none of the tools provide support services, alternative input options, audio, media, animation and tactile outcomes. None of the tools has an input/output alternative, special considerations for accessibility adjustments, and performance in closed systems. Almost half of the tools have a score of 0,327 in General guidelines such as user preference settings. 96% of the tools have the same score in Online documentation and help. Furthermore, all the tools have a low score in Compatibility with assistive technology and in Pointing devices. Some improvements can also be done to current RE tools to enhance Internationalization: adaptation capabilities through use of user profiles and group profiles, use in a cultural diversity and multilingual context and appropriate formats, units, and currency for international audience. In contrast, most of the RE tools offers web-based documentation, help, FAQs and forums in multiple languages.

In summary, both internationalisation and accessibility are poorly supported by current RE tools and the margin for amelioration is large. This is evidently an important drawback for globally-distributed projects, since internationalisation features are needed to ease individuals' work, no matter where they come from. On the other hand, software companies are more and more concerned about making applications accessible to all people, whether they have a disability or not, and RE tools should not be exceptions. Figure 2 shows that Top Team Analyst come first in the tools' classification even if it has lower i18n score than Rational DOORS.

Figure 1 shows the box plots of scores according to Internationalization, General Guidelines, Inputs, Outputs and Documentation. In most cases, the median scores obtained for each capability are low, although the median score of the general guidelines is slightly bigger than that of the others. This suggests that basic functionality is generally implemented, at least for the sample of tools analyzed. Notice that the dispersion of scores is substantially higher for i18n, suggesting that the implementation of this capability in the RE tools is more heterogeneous compared to the others capabilities, so developers should analyze

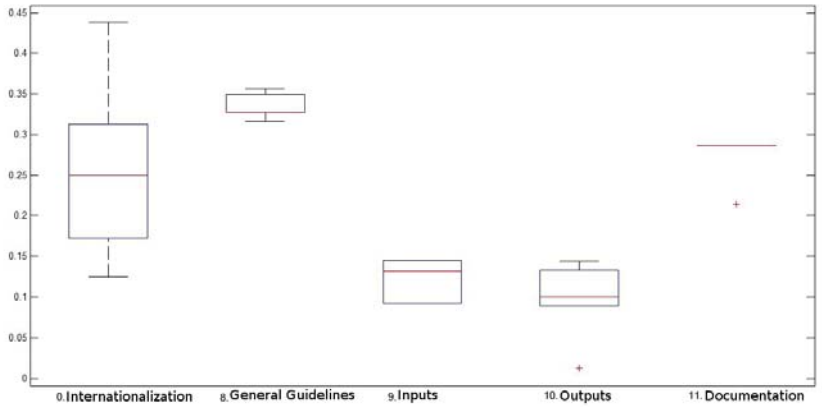


Fig. 1. Boxplots diagram of tools' scores

i18n capabilities in detail when purchasing a tool for GSD. In contrast, Documentation presents a very low degree of dispersion, which suggests uniformity in the RE tools online documentation.

GSD has been revealed over the last years as an important trend in software development, so it is expected a growing demand of organisations towards internationalised RE tools. However, the development of accessible tools is expensive and the market of these solutions is much smaller, thus the return of investment is not guaranteed. Therefore, there is a need for accessibility resources that help RE tools' developers to build accessible applications more easily and reducing development costs, for instance by means of reusable domain catalogues [24], accessibility architectures and implementation APIs.

5.1 Threats to validity

Conclusion validity Detailed information regarding the databases used is presented in order to show reasonable results which can be reproduced by other research groups. The Cohen's Kappa coefficient was used to calculate interrater agreement among the researchers in the evaluation. The Kappa coefficient was 0.91 ($p < 0.0001$), 95% CI (0.85, 0.97). The good strength of agreement points out that the data obtained seem to be trustworthy thus reducing this threat to validity. If the researchers disagreed on including or excluding a tool, this was discussed until agreement was obtained. The p-value and confidence interval were calculated using the method of Fleiss.

Construct validity In order to obtain as complete a set of tools related to the given research topic as possible, the search term was derived systematically. We have searched broadly in general databases which index most well reputed tools fora. The level of confidence achieved was raised by using the triangulation

technique in the cross-check, in order to enable the researchers to draw valid conclusions. We selected the evaluation of accessibility aspects of three vendors based on previous research [13], three researchers supervised the evaluation work, and other three researchers reviewed their findings.

Internal validity The selection of tools found during the cross-check was performed by one researcher only, which may have resulted in missing tools. However, the capabilities of 13 RE tools were meticulously assessed by the authors, three software engineers with experience in the standard-based tools assessment [7] [13] and audit [21], and cross-checked against the vendors' evaluation of their tools. A high conformance value was achieved thus reducing this threat.

External validity The results of our study were considered with respect to approaches in the software domain. Thus, the classification presented and the conclusions drawn are only valid in the given context. Therefore, this classification can serve as a starting point for new research on accessibility and i18n in RE tools. Additional tools that are identified in the future can be assessed using the procedure followed during this study. The conclusions with respect to the defined research questions may be true independently of this threat.

6 Conclusion

In this paper, research on i18n and accessibility regarding RE tools for GSD settings has been presented. An evaluation was conducted involving 13 RE tools obtained from a set of relevant databases. I18n and accessibility capabilities in order to facilitate the stakeholders' work on GSD projects were extracted from international standards and technical reports: ISO/IEC 9241 (part 171), ISO/IEC 24751 (part 2), ISO/IEC 9241 (part 151), CWA 14928 and ISO/IEC TR 24766:2009 TR. After analyzing the results of the evaluation, our findings showed that there is a lack of tool adequacy to the particular needs of accessibility and i18n for RE tools.

In future work, it is intended to enrich the questionnaire to address the multi-cultural challenges in the GSD, since a need of focusing the RE tools industry attention on GSD capabilities has been shown. We will also extend the scope of the study to cover other accessibility standards and guidelines such as: WCAG 2.0 and CWA 15554:2006 for web versions of RE tools, IBM Software Accessibility Checklist and ISO TS 16071. Web accessibility evaluation tools listed in the Web Accessibility Initiative (WAI) home can also be considered to automatically check some aspects of web accessibility in RE tools. Moreover, the use a higher sample size for decreasing the bias of the estimates will be considered.

Acknowledgments. This work has been funded by the PEGASO/PANGEA project (TIN2009-13718-C02-02) and by the Mediterranean Office for Youth (MOY).

References

1. CEN-Learning Technologies Standards Observatory Contents. Available in http://www.cen-ltso.net/Main.aspx?pdf=-1. last access: April 2012
2. IEEE P1484. Computer Managed Instruction. IEEE Learning Technology Standards Committee. Available in http://www.ieeeltsc.org. Last access: April 2012
3. W3C Internationalization Activity. http://www.w3.org/International/. Last Access: April 2012
4. ISO/TS 16071 Ergonomics of human-system interaction – Guidance on accessibility for human-computer interfaces (2003)
5. Basili, V.R., Shull, F., Lanubile, F.: Building knowledge through families of experiments. IEEE Trans. Softw. Eng. 25, 456–473 (July 1999)
6. Basili, V., Rombach, H.: The TAME project: towards improvement-oriented software environments. IEEE Trans. Softw. Eng. 14, 758–773 (1988)
7. Carrión, I., Fernández-Alemán, J.L., Álvarez, J.A.T.: Usable privacy and security in personal health records. In: INTERACT (4). pp. 36–43 (2011)
8. Cheng, B.H.C., Atlee, J.M.: Research directions in requirements engineering. In: FOSE '07: 2007 Future of Software Engineering. pp. 285–303 (2007)
9. Damian, D., Lanubile, F., Oppenheimer, H.L.: Addressing the challenges of software industry globalization: the workshop on global software development. In: Proc. of ICSE. pp. 793–794 (2003)
10. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretization of continuous features. In: Proc. of ICML. pp. 194–202 (1995)
11. Ebert, C., Hernandez Parro, C., Suttels, R., Kolarczyk, H.: Improving validation activities in a global software development. In: Proc. of ICSE. pp. 545–554 (2001)
12. Ergonomics of human-system interaction. Guidance on World Wide Web user interfaces: ISO/IEC 9241-151 (2008)
13. Carrillo de Gea, J.M., Nicolás, J., Fernández Alemán, J.L., Toval, A., Ebert, C., Vizcaíno, A.: Requirements engineering tools. IEEE Softw. 28(4), 86–91 (2011)
14. Carrillo de Gea, J.M., Nicolás, J., Fernández Alemán, J.L., Toval, A., Ebert, C., Vizcaíno, A.: Requirements engineering tools: Capabilities, survey and assessment. Information and Software Technology (In Press) (2012)
15. Graf, S., List, B.: An evaluation of open source e-learning platforms stressing adaptation issues. In: Proc. of ICALT. pp. 163–165 (2005)
16. He, Z., Bustard, D.W., Liu, X.: Software internationalisation and localisation: practice and evolution. In: Proc. of PPPJ/IRE. pp. 89–94 (2002)
17. Hinds, P., Bailey, D.E.: Understanding conflict in distributed teams. Organization Science 14(6), 615–632 (2003)
18. Information technology – Individualized adaptability and accessibility in e-learning, education and training –: ISO/IEC 24751. ISO (1996)
19. ISO/IEC JTC 1 SC 7: ISO/IEC TR 24766:2009. Geneva, Switzerland, 1st edn.
20. Kitchenham, B.: DESMET: a method for evaluating software engineering methods and tools. Tech. Rep. TR96-09, Dept. of Computer Science, University of Keele (1996)
21. Martínez, M.A., Lasheras, J., Fernández-Medina, E., Toval, A., Piattini, M.: A personal data audit method through requirements engineering. Comput. Stand. Interfaces 32(4), 166–178 (Jun 2010)
22. Mistrík, I., Grundy, J., Hoek, A., Whitehead, J.: Collaborative software engineering: challenges and prospects. In: Mistrík, I., Grundy, J., Hoek, A., Whitehead, J. (eds.) Collaborative Software Engineering, pp. 389–403. Springer Berlin (2010)

23. Technical Committee ISO/TC 159, Ergonomics, Subcommittee SC 4, Ergonomics of human-system interaction: ISO/IEC 9241-171 (2008)
24. Toval, A., de Gea, J.M.C., Nicolas, J., Fernandez-Aleman, J.L., Toval, R.: Learning systems development using reusable standard-based requirements catalogs. In: EDUCON (2011)

A Appendix

Table 7. Results summary

Category	Aligned Elements	Bright Green Projects	QFD capture	Ra Quest	Rational DOORS	Req Man	RTIME	RM Trak	Test Track RM	Top Team Analyst	Trace Cloud	Track Studio	Visible Thread	Average
0. Internationalization														
0. Internacionalizacin	0,250	0,313	0,125	0,125	0,438	0,250	0,125	0,188	0,250	0,313	0,313	0,313	0,188	0,245
Average	0,250	0,313	0,125	0,125	0,438	0,250	0,125	0,188	0,250	0,313	0,313	0,313	0,188	0,245
8. General guidelines														
8.1. Input/output alternatives	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
8.2. Names of user interface elements	0,813	0,875	0,813	0,813	0,875	0,875	0,813	0,813	0,813	0,813	0,875	0,875	0,875	0,841
8.3. User preference settings	0,143	0,071	0,143	0,286	0,000	0,000	0,214	0,071	0,286	0,286	0,000	0,000	0,000	0,115
8.4. Special considerations for accessibility adjustments	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
8.5. General control an operation guidelines	0,650	0,700	0,600	0,650	0,650	0,650	0,650	0,650	0,650	0,650	0,650	0,600	0,650	0,646
8.6. Compatibility with assistive technology	0,208	0,208	0,208	0,208	0,208	0,208	0,208	0,208	0,208	0,208	0,208	0,208	0,208	0,208
8.7. Closed systems	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
Average	0,337	0,347	0,327	0,357	0,327	0,327	0,347	0,327	0,357	0,357	0,327	0,316	0,327	0,337
9. Inputs														
9.1. Alternative input options	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
9.2. Keyboard focus	0,667	0,667	0,667	0,667	0,333	0,333	0,667	0,667	0,667	0,667	0,333	0,333	0,333	0,538
9.3. Keyboard input	0,176	0,118	0,176	0,176	0,118	0,118	0,176	0,147	0,176	0,176	0,118	0,118	0,118	0,147
9.4. Pointing devices	0,036	0,036	0,036	0,036	0,036	0,036	0,036	0,036	0,036	0,036	0,036	0,036	0,036	0,036
Average	0,145	0,118	0,145	0,145	0,092	0,092	0,145	0,132	0,145	0,145	0,092	0,092	0,092	0,121
10. Outputs														
10.1. General output guidelines	0,333	0,333	0,000	0,333	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,077
10.2. Visual output (displays)	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
10.3. Text/Fonts	0,000	0,000	0,250	0,000	0,000	0,000	0,250	0,000	1,000	0,750	0,000	0,000	0,000	0,173
10.4. Color	0,000	0,200	0,200	0,200	0,000	0,000	0,200	0,200	0,000	0,200	0,000	0,000	0,000	0,092
10.5. Window appearance and behavior	0,400	0,400	0,400	0,400	0,400	0,400	0,400	0,350	0,400	0,400	0,400	0,400	0,400	0,396
10.6. Audio output	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
10.7. Text equivalents of audio (captions)	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
10.8. Media and animation	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
10.9. Tactile output	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
Average	0,111	0,133	0,122	0,133	0,089	0,089	0,122	0,100	0,133	0,144	0,089	0,089	0,089	0,111
11. Online documentation, help and support services														
11.1. Documentation and Help	0,400	0,400	0,400	0,400	0,400	0,400	0,400	0,300	0,400	0,400	0,400	0,400	0,400	0,392
11.2. Support services	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
Average	0,286	0,286	0,286	0,286	0,286	0,286	0,286	0,214	0,286	0,286	0,286	0,286	0,286	0,280
Total Average	0,226	0,239	0,201	0,209	0,246	0,209	0,205	0,192	0,234	0,249	0,221	0,219	0,196	0,219

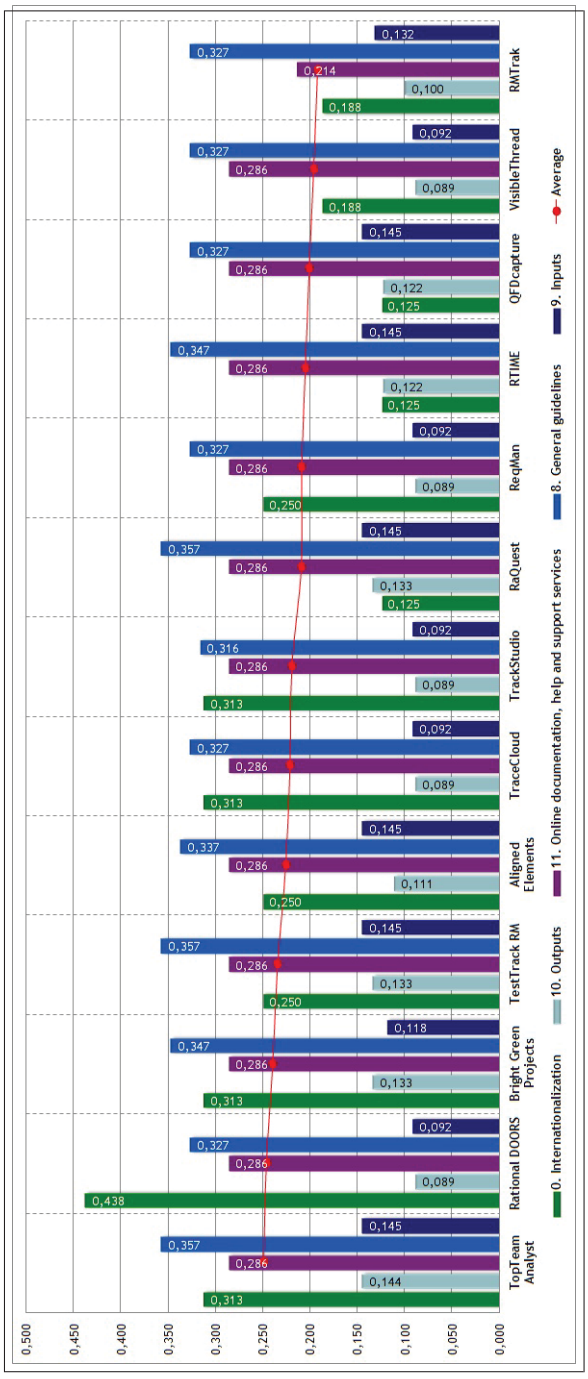


Fig. 2. Tools' scores per categories

Using Small Affordable Robots for Hybrid Simulation of Wireless Data Access Systems

Gorka Guerrero, Roberto Yus, and Eduardo Mena

IIS Department, University of Zaragoza
María de Luna 1, 50018, Zaragoza, Spain
{524080,ryus,emena}@unizar.es

Abstract. During the last years, research on data processing in wireless environments has increased due to the emergence of mobile devices that are able to obtain real environmental sensory information (e.g., smartphones). Testing the different approaches in a real environment is not always possible due to high costs of deployment of hardware and users in many cases. However, the real world complexity can be simplified according to our needs as information systems always deal with simplified abstractions of real objects. For example, a system considering the location of a real car could simplify it as a certain entity with the same movement path. This can be achieved by using software simulations, which obtain approximated results reducing the costs. Nevertheless, it is difficult to develop an accurate real-world model to simulate the environmental conditions (e.g. uneven tracks, dynamic wireless network coverage, etc.).
We introduce in this paper a hybrid simulation platform that is able to recreate real-world scenarios more accurately than software simulations. For that, it uses small affordable robots equipped with sensors in controlled real environments as counterparts of real moving objects and the scenario where they are involved. This enables testing the system considering real communication delays, real sensor readings, etc., instead of having to simulate such events. Finally, we present the experimental evaluation of the system using LEGO Mindstorms robots to simulate a real rowing race at the San Sebastian bay.

Keywords: Hybrid Simulation, Wireless Data Access, LEGO Mindstorms

1 Introduction

Nowadays there exist more and more mobile devices, such as smartphones, tablets, or laptops with different types of sensors integrated. Because of their small size, they can be attached to moving objects (e.g. cars, people, etc.) and thanks to their communication mechanisms (Wi-Fi, Bluetooth, 3G, etc.), they are suitable for remote sensing. Thus, research in fields such as mobile computing or data mining are considering these devices to develop systems to manage sensory information obtained using wireless communications.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 693-706, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

To validate these systems researchers must perform tests, but performing real tests is not always possible. For example, testing a scenario where there exist a great number of cars and people moving around a city have high costs of deployment. However, from the point of view of a data access system the complexity of the real world can be simplified according to the information it wants to obtain. In the previous example, if the data access system is interested in the location of the cars and people, they can be considered as simpler entities with the same movement path. For this reason, researchers use software simulation to test their approaches that dramatically reduce the testing costs. The main problem with software simulations is that the fidelity of their results has been a concern. Obtaining a real-world model to simulate environmental conditions as wireless communication delays or disconnections, sensor failures, etc., is a real challenge.

We propose in this paper a system to carry out hybrid simulations of data access systems, which is able to recreate some real-world scenarios more accurately than software simulations. Real objects, which are abstracted in software simulations, are simplified in our system by using real moving objects with similar characteristics. For this, we use small affordable robots equipped with real sensors in controlled real environments. Using these robots we are able to test the system considering real aspects that are difficult to simulate, such as communication delays, unexpected events, and sensor accuracy. For example, considering a system that analyzes in real-time the sensory data acquired in a sport event where involved moving objects are equipped with location sensors, video cameras, and wireless communications, generate a real-world model to simulate the environmental conditions is a challenging task. However, using our system some elements will be simulated (e.g., the scale of the scenario) while the real hardware employed leads to obtain real environmental conditions.

The rest of the paper is organized as follows. In Section 2 we present the motivation and technological context behind our system. In Section 3 we present the system architecture. In Section 4 and Section 5 we present how moving objects and the base station are modeled in our system, respectively. In Section 6 we present a prototype developed to test our system. In Section 7 we review some interesting related works. Finally, in Section 8 we present our conclusions and future work.

2 Context

In this section we explain a sample use case to motivate the development of a hybrid simulator. Then, we will explain the technological context of our system.

2.1 Motivation

The use case we are considering as example of wireless data access system [13] deals with the acquisition of sensory information (i.e. location, camera video,

etc.) of a sport event, the rowing race of San Sebastian (Spain). With this information, the system is able to help the technical director (in charge of the live TV broadcasting of the race) to select the best camera views.

A software simulator was used to test this system fed with the GPS locations of all the rowing boats obtained every second during a race. However, one of the problems of this test was that, even then the GPS locations were real, other sensor information (for example, the camera video streams) was simulated. Moreover, to test a different rowing boat configuration (for example, changing their movement path) new GPS traces are needed and obtaining them is difficult as the race takes place once a year.

In Table 1 we compare three different approaches to test a data access system (real test, software simulation, and our system), according to different criteria: a) realistic communication delay, how accurate is the behavior of the communications?, b) sensors used, how realistic are the sensors considered? , c) scenario, how much space is required to test the scenario?, d) repeatability, is easy to repeat the test?, e) interaction with the environment, how accurate are the environmental conditions?, f) time lapse, how much time is required to perform the test?, g) cost, how high is the cost to perform the test?.

Dealing with	**Real test**	**Simulation**	**Our system**
Realistic communication delay	✓✓	✗	✓
Sensors used	✓✓	✗	✓
Scenario	✗	✓✓	✓
Repeatability	✗	✓	✓
Interaction with the environment	✓	✗✗	✓
Time lapse	✗	✓✓	✓
Cost	✗	✓✓	✓

Table 1. Comparing the different approaches to test a data access system: using real tests, software simulations, and our system.

Compared with a software simulation, our hybrid simulation provides real communication delays (as it uses real communication mechanisms), real sensors (that obtain real data), and real environmental conditions. However, the duration of the test in our system is higher than in a software simulation (for example, because of the maximum sample acquisition rate of the sensors), and the cost of deployment could be also slightly higher. Compared with a real test, our hybrid simulation requires less space and infrastructure for the scenario (as it can be scaled), it is easier to repeat the tests (for example to obtain the results with new restrictions), the duration of the test could be reduced, and the cost is much smaller.

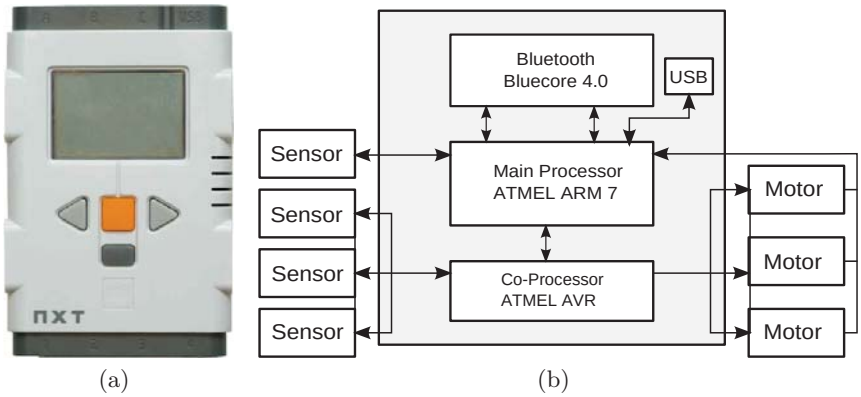
Therefore, our system allows to perform hybrid simulations of different scenarios where moving objects equipped with sensors capture and store data. To configure a specific scenario, the user selects its location and scale as well as the

amount of moving objects involved. For each object in the scenario, we have to specify its sensors and the path or movements that it is going to follow.

2.2 Technological Context

We have decided to use LEGO Mindstorms [2] robots in the prototype of our proposed system to represent real moving objects (see Section 6). LEGO Mindstorms is a basic but powerful, flexible, and affordable robot kit that was initially designed as an advanced toy, but is widely used to develop valuable designs [6] and in academic environments [4, 7, 9].

Architecture: As we can see in Figure 1, the NXT brick has a main processor and a co-processor. First, an Atmel 32-bits ARM processor is mainly used to control communication with Ultrasonic sensors, and USB/Bluetooth communication mechanisms. An Atmel 8-bits AVR processor controls the rest of sensors and servo motors connected to the NXT. Before starting to code programs for the NXT, we have to consider the limited capabilities of the robot's hardware, since we have only 256KB of FLASH memory and 64KB of RAM inside the 32-bit ARM processor, so the program has to be optimized to run on it. Each NXT can have attached up to 4 sensors and 3 motors to the available ports. In addition, the NXT is equipped with USB and Bluetooth 4.0 communications.



(a) (b)

Fig. 1. The NXT brick (a) and its main components (b).

Sensors: The Most popular LEGO Mindstorms sensors are the following:

- *Ultrasonic*, that returns the distance to an obstacle.
- *Touch*, that detects collisions.
- *Sound*, that measures sound levels.

– *Light/Color*, that detects different levels of grey or color scale.

A complete description of all sensors with their characteristics, schema and operations is available in the *LEGO MINDSTORMS NXT Hardware Developer Kit*[1]. However, the LEGO NXT brick is an Open Source hardware system and so we can easily find third-party sensors. For example, Hi-Technics[2], Mindsensors[3], and Dexter Industries[4] are some of the companies that distribute third-party sensors such as:

– *Compass*, that returns the direction.
– *3-Axis Accelerometer*, that measures the acceleration in three axis.
– *Gyro*, that measures the number of degrees per second of rotation.
– *Magnetic*, that can detect magnetic fields.

Figure 2 shows some of these sensors. All of them use the same interface to be connected to the NXT, a 6-position modular connector that features both analog and digital interfaces. However, the NXT brick is not limited to these sensors, for example, we can attach an external GPS or a camera via Bluetooth [11].

Fig. 2. Some of the available sensors for the NXT.

Software: NXT-G is the standard programming software for NXT and is included in the kit. NXT-G is based on LabVIEW graphical programming software, created by National Instruments and released in December 2006. Nevertheless, LEGO has released the firmware of the NXT as Open Source, so we can install other firmware on the NXT and create more complex programs for the robot using other popular programming languages. For example, RobotC [10], which is based on C, is distributed as Proprietary Commercial Software, and leJOS NXJ [1], a high-level Open Source programming language based on Java, that is the firmware we have chosen for NXTs in our prototype.

[1] `http://mindstorms.lego.com/eng/Overview/nxtreme.aspx`
[2] `http://www.hitechnic.com/`
[3] `http://mindsensors.com/`
[4] `http://dexterindustries.com/`

3 System Architecture

Our system consists of two main parts shown in Figure 3: 1) the selected moving object responsible for collecting data from all its attached sensors, and 2) the base station which is in charge of receiving data from all the moving objects and controlling them. This architecture allows the system to connect different moving objects, that are used to represent real objects (e.g., cars, rowing boats, people, etc.). Also the base station allows the user to access the data of the sensors in real-time and control the robots.

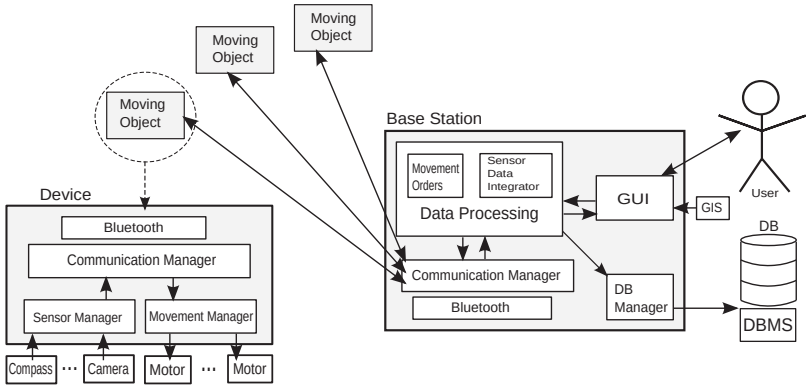


Fig. 3. System Architecture.

Therefore, the equivalence between performing tests in a real scenario and using our hybrid simulation system can be summarized as follows. We use LEGO Mindstorms robots as simplified abstractions of the real moving objects in a real scenario, as they can be controlled to perform similar movements. Instead of using certain real sensors, robots are equipped with (real) LEGO sensors that can perform nearly the same tasks and acquire the same type of information than we want to test. Finally, sometimes we do not need to test the system in the same scenario; instead we can deploy a simplified version of the real scenario (for example, at a smaller scale).

4 Simulating Moving Objects

The moving objects that our system manages are a simplified version of real objects to simulate, and the system controls them during the hybrid simulation to behave as the real ones. Thanks to this, the system is able to capture real data from the environment as real moving objects could do. To perform these tasks, our moving object has three main modules that we explain in the following.

4.1 Sensor and Movement Managers

As we have explained in Section 2.2, we can manage different types of sensors for our selected moving objects, so we need a module that can handle them and that allow us to add new sensors easily. Each sensor has different control options, different access functions, and provides different data, so the module has to deal with this heterogeneity.

We have analyzed several of the available sensors (e.g. sound, light, compass, accelerometer, etc.) to be integrated in our moving object, their purpose and the data they generate. We have noticed that, among the data provided by the sensors, we can get integer or float numbers (depending if we are measuring the distance to an obstacle or the angular velocity of a turn taken by the moving object). Also, we can receive different number of parameters from sensors too. For example GPS returns latitude and longitude data, an accelerometer returns the acceleration in the X, Y, and Z axis, and a compass only returns the direction.

The movement manager module receives orders from the base station and decodes them to send movement orders to the proper motor. Each of the motors are independent and can perform two different movements: backwards and frontwards. By combining these movements and by using different motors, the movement manager module is able to recreate the behavior of real-world moving objects.

4.2 Communication Manager

Once we have the data from the Sensor Manager, the Communication Manager module is in charge of sending it to the base station. We have developed a communication protocol to send all this different information from the sensors to the base station.

The protocol consists basically on a message generated by using the code of the NXT, the code of the sensor, and the data we want to send. The quantity of data depends on the sensor, so we have to separate each parameter while making the message easy to be parsed on the base station side (Table 2 shows some sample sensor messages of the communication protocol).

Compass Sensor	NXT code	Sensor Code	Direction		
GPS Sensor	NXT code	Sensor Code	Latitude	Longitude	
Accelerometer Sensor	NXT code	Sensor Code	X-Axis	Y-Axis	Z-Axis

Table 2. Sample of communication messages used by the system.

Once the message has been generated, the communication manager has to send it to the base station. The communications this module manages are built over a Bluetooth layer (Bluetooth is the default communication mechanism in

NXT brick and leJOS includes Bluetooth libraries in its API), but we could use other communication mechanisms as Wi-Fi or ZigBee [5].

Due to Bluetooth limitations, the maximum number of active slaved devices paired to the master device (base station) in a piconet is seven. This number may be higher if the implementation handles parked connections. If we have inactive or parked devices, we can connect up to 255 devices, which the master device can bring into active status at any time.

5 Base Station Side Application

The other main component of our system is the application on the base station side. The purpose of this program is to collect all data sent by moving objects connected to it and display this information in the Graphical User Interface (GUI). Also, it enables the user to control the moving objects. For all these tasks, the program uses different modules, explained in the following.

- *Graphical User Interface*
  The GUI is an important part of the hybrid simulator because it needs to ease the user interaction. All data that the moving objects capture and send to the base station have to be shown in the GUI according to their types. For example, for some data types such as the location, it is necessary not only to show the last data received, but also a history with all samples received from the beginning. Moreover, the GUI also has to be capable to show to the user in real time all the video streaming of the cameras. Therefore, the GUI module uses tables, graphs, and external Geographical Information Systems (GIS) to display the information easy and effectively.
- *Communication Manager*
  This module is very similar to the Communication Manager module explained in Section 4.2. It is in charge of receiving all the messages from the moving objects and also of sending them control messages. This module is built over a Bluetooth layer too, but this time the layer depends on the Operating System installed. In our prototype we have used Bluecove[5] as Bluetooth Java library that interfaces with Microsoft Bluetooth stack in Windows (XP and Vista) and also with Mac OS X.
- *Data Processing*
  This module processes data received from the Communication Manager. First, all data is parsed and processed to be treated and displayed according to the source. As in the communication protocol shown in Table 2, the message received contains the code of the moving object and the sensor, so when we decode it, we have to send the information to the GUI. Also, this information will be stored in a database to keep all the information received to be analyzed later if it is necessary.
  This module also manages all the robot movement orders they have to perform. From the GUI, we select the desired order and this module generates

[5] `http://bluecove.org/`

the message that the Communication Manager module will send to the moving object. In addition, this module controls also all the camera movements so, we can control remotely what each camera is viewing.

– *Database Manager*
Although all the information received from the moving objects is shown in the GUI, it is also interesting to store it to be analyzed later. This could be interesting because occasional anomalies can occur during the test (for example, a sensor could obtain a wrong measurement) and it is difficult to realize on them in real-time. So, storing the data in a database allow us to analyze this information.

6 Experimental Evaluation

We have developed a prototype of the system proposed to perform the hybrid simulation of the scenario explained in Section 2.1. Figure 4 shows the GUI of the prototype where three main areas can be observed:

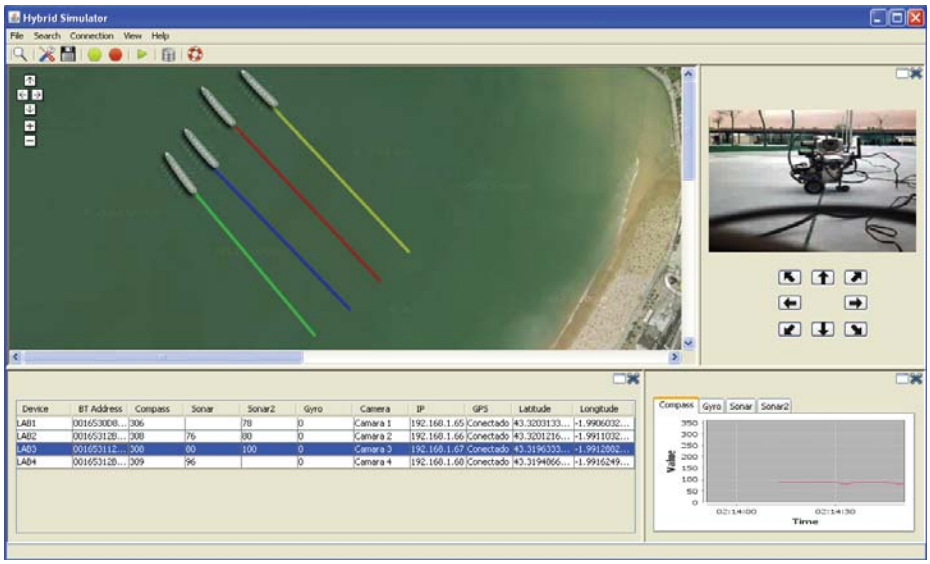


Fig. 4. Graphical User Interface of the prototype developed.

– *Map section*, where the real-time location of the simulated moving objects is shown using Google Maps[6] as GIS.

[6] `http://maps.google.com/`

– *Camera section*, that shows the video-stream of the cameras and enables the user to control them.
– *Sensory Information section*, where data obtained by sensors is shown both in a tabular way and using charts.

In our hybrid simulation we use four LEGO Mindstorms robots to simulate the rowing boats, with a software application (developed using leJOS v0.91[7]) that implements the ideas explained in Section 4. Each robot is equipped with a Bluetooth SysOnChip GPS and an Edimax IP camera. Moreover, we have attached also to the robots a compass, a gyro, and ultrasonic sensors to obtain additional information. The configuration of the robots is shown in Figure 5, notice that all the connection wires have been removed for the picture.

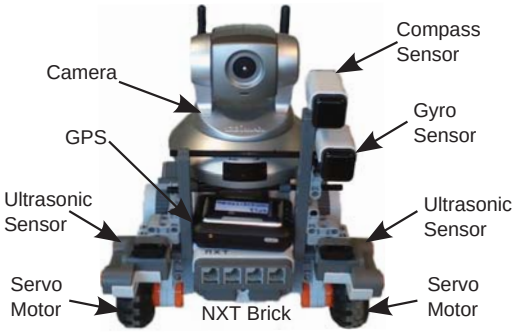


Fig. 5. LEGO Mindstorms configuration used in our test.

In the real rowing race there are four rowing boats involved, each of them is equipped with a GPS transmitter and a video camera. The boats leave from the start line, they sail 1.5 miles, and they come back after making a turn in the sea. To recreate the movement path of the rowing boats, each robot has been programmed to move straight and back to the beginning after performing a 180° turn. The different speeds of the boats have been simulated by adjusting the robots to experience random speed changes during the test (depending on the part of the race). With this configuration, Figure 6 shows a picture of the start of the real rowing race and the same moment in our test.

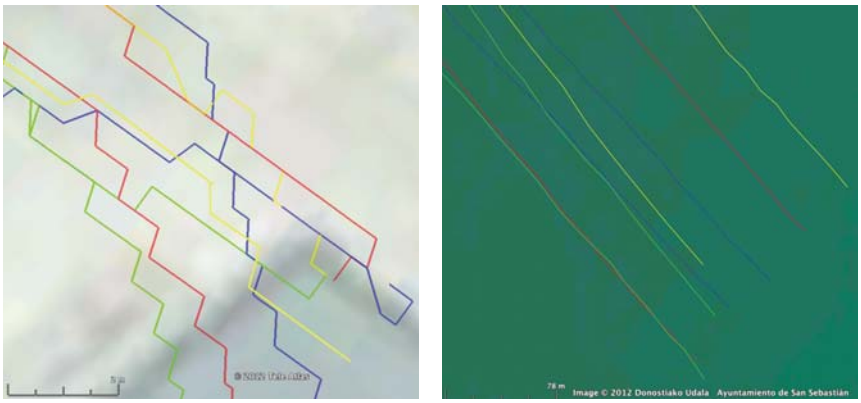
One of the most interesting sensory information in this scenario is the location of the boats, as it can be used for example to check the distance between them. In our tests, the robots obtain their location every second from the GPS and send this information to the base station. Then, the base station translates these locations to the simulated bay of San Sebastian by scaling it to be represented over the map. Figure 7 shows part of the path followed by the robots in our

[7] `http://lejos.sourceforge.net/`

(a) (b)

Fig. 6. Image from the real scenario (a) and image from our test (b).

scaled scenario and the translation to the rowing race scenario[8]. Notice that the GPS accuracy (around 2 meters) makes the paths in Figure 7(a) to cross each other and do not follow a straight line. This is because the distance between the robots was around 20 cm, and a imprecise measurement of the GPS (e.g., 1 meter) in the location of a robot makes it to appear outside its lane. To deal with this, in the translation of Figure 7(b) we have scaled the distance between the robots and the distance they travel each second. Then, we have applied the same error (at maximum was 2 meters) that the GPS obtained at each sample. Notice that the GPS error in Figure 7(b) seems smaller and that is because of the scale of the image.

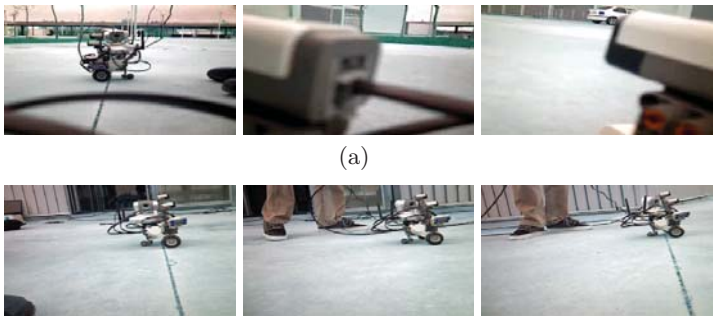


(a) (b)

Fig. 7. Location obtained by the GPS of the robots during the test (a), and the translation to the simulated scenario (b).

[8] The represented part corresponds to the start/finish of the race; so, for each boat with the same color, it shows its leaving of the harbor and its returning.

Another interesting sensory information is the video stream that the cameras are capturing. We have recorded the whole event and we present in Figure 8 some interesting captures. The captures are obtained for the same time instants for the cameras on board "Robot2", Figure 8(a), and "Robot3", Figure 8(b). Notice that at the first time instant both cameras are recording their adjacent robots, so in the real scenario both cameras would be recording the adjacent boats. Camera from "Robot2" is viewing its adjacent robot, "Robot3", and camera from "Robot3" is viewing its adjacent robot, "Robot2". The cameras we used in our tests can be remotely controlled as the real cameras in the race; so, we rotated the camera on board "Robot2" horizontally to the left during the test, and the effect of this rotation and the different speeds of the robots can be observed in the rest of the captures. The camera on board "Robot3" was not rotated and so, it views "Robot2" during the rest of the captures.



(a)

(b)

Fig. 8. Real captures of Robot2 camera, (a), and Robot3 camera, (b), during a test.

Analyzing the results of the test, the prototype could be useful to train technical directors offline. A technical director must make quick decisions in a live broadcasting to select the camera whose view must be broadcasted. A trained technical director is able to make right decisions in such a scenario, but he/she have to be prepared for unexpected events (for example, a boat crash). Therefore, our prototype could be used to simulate a rowing race and the robots can be controlled to randomly stop or the cameras can be remotely turned off.

7 Related Work

The use of small robots to simulate the behavior of real moving objects has been also considered when testing vehicle scenarios. For example, in [12] they present a system to test Plug-in Hybrid Electric Vehicle scenarios using robots. Their testbed only take into account these scenarios, but we share some common points as the development of a GUI that easily helps the user to analyze the information captured by the robots. In [8] they also use small robots to test real scenarios.

This work is focus on RFID communications between vehicles, so, they use real RFID equipped robots. They program the robots to reproduce real scenarios as we do, but they are not interested in sensory information that the robots could provide.

The term hybrid simulation has been previously used in [3]. In this work, the authors remark that "*the fidelity of simulation results has been a concern*", and that a platform that improves software simulations is needed. Therefore, they presented a hybrid simulation platform used for wireless application/protocol testing, which combines software simulations with real hardware for communication. While we are concerned about moving objects and data access systems, we share the idea of using real hardware to increase the fidelity of the testing of systems.

As we previously commented, LEGO Mindstorms robots are widely used in academic environments thanks to their affordable costs and the ease of their programming. In [6] they explain the features that make LEGO Mindstorms a suitable platform for college students. Among the many works that use LEGO Mindstorms for teaching, in [7] the authors show the benefits of using these robots for teaching introductory programming. In [4], LEGO Mindstorms are used to teach C language in a more appealing way for students. Finally, in [9] the authors present a practical robotics engineering course using LEGO Mindstorms instead of advanced robots. They explain the results of their experience with the students showing that using this affordable robots is enough to introduce them to robotics typical challenges as computer vision, autonomous navigation, etc.

8 Conclusions and Future Work

We have presented an approach for hybrid simulation of wireless data access systems based on the use of small affordable robots. The combination of real hardware with the simulation of the movements or the scenarios, allows our system to increase the fidelity of the tests performed for a wireless data access system compared to the use of software simulations. Therefore our system presents the following benefits:

- It allows testing in simple but real scenarios as a scaled version of the real ones. Therefore, the scenarios will require less area to be deployed and less infrastructure.
- It uses real sensors and communication mechanisms, so wireless data access systems can be tested in environments with real failures, accuracy, and delays.

Our system allows to perform hybrid simulations of different scenarios where moving objects equipped with sensors capture and store data. To configure a specific scenario, the user selects its location and scale as well as the sensors and movement paths for each moving object involved. To improve Bluetooth limitations, as future work, we plan to extend the prototype considering Wi-Fi and ZigBee communications between the robots and the base station. We also plan

to improve the system by adapting the behavior of the robots to autonomously complete predefined tasks in a collaborative approach.

Acknowledgments. This research work has been supported by the CICYT project TIN2010-21387-C02-02 and DGA-FSE.

References

1. B. Bagnall. *Intelligence Unleashed: Creating LEGO NXT Robots With Java*. Variant Press, 2011.
2. D. Baum. *Dave Baum's Definitive Guide to LEGO Mindstorms*. APress L. P., 1st edition, 1999.
3. P. De, A. Raniwala, S. Sharma, and T. Chiueh. Mint: a miniaturized network testbed for mobile wireless research. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2731 – 2742 vol. 4, March 2005.
4. S. H. Kim and J. W. Jeon. Educating C language using LEGO Mindstorms robotic invention system 2.0. In *IEEE International Conference on Robotics and Automation. ICRA 2006*, pages 715 –720, May 2006.
5. P. Kinney. Zigbee technology: Wireless control that simply works. *ZigBee Alliance*, 2003.
6. F. Klassner and S. Anderson. LEGO Mindstorms: not just for k-12 anymore. *Robotics Automation Magazine, IEEE*, 10(2):12 – 18, June 2003.
7. P. B. Lawhead, M. E. Duncan, C. G. Bland, M. Goldweber, M. Schep, D. J. Barnes, and R. G. Hollingsworth. A road map for teaching introductory programming using LEGO Mindstorms robots. In *Working group reports from ITiCSE on Innovation and Technology in Computer Science Education*, ITiCSE-WGR '02, pages 191–201, 2002.
8. J. Munilla, A. Ortiz, and A. Peinado. Robotic vehicles to simulate rfid-based vehicular ad hoc networks. In *3rd International ICST Conference on Simulation Tools and Techniques*, SIMUTools '10, pages 49:1–49:2, 2010.
9. A. C. Murillo, A. R. Mosteo, J. A. Castellanos, and L. Montano. A practical mobile robotics engineering course using LEGO Mindstorms. In *Research and Education in Robotics - EUROBOT 2011*, volume 161 of *Communications in Computer and Information Science*, pages 221–235. Springer Berlin Heidelberg, 2011.
10. A. W. Schueller. *Programming with Robots*. Available online: http://carrot.whitman.edu/Robots/notes.pdf.
11. D. E. Stevenson and J. D. Schwarzmeier. Building an autonomous vehicle by integrating LEGO Mindstorms and a web cam. In *38th SIGCSE technical symposium on Computer science education*, SIGCSE '07, pages 165–169, 2007.
12. W. Z. W. Su and M.-Y. Chow. A digital testbed for a PHEV/PEV enabled parking lot in a smart grid environment. In *Innovative Smart Grid Technologies (ISGT 2012)*, 2012.
13. R. Yus, E. Mena, J. Bernad, S. Ilarri, and A. Illarramendi. Location-aware system based on a dynamic 3D model to help in live broadcasting of sport events. In *19th ACM International Conference on Multimedia (ACMMM 2011)*, pages 1005–1008, November 2011.

Agile Moodle: Una plataforma para el Aprendizaje Ágil en Ingeniería del Software

Pablo Ortíz[1], Jennifer Pérez[2], Santiago Alonso[2], José Luis Sánchez[2], Javier Gil[2]
Universidad Politécnica de Madrid
E.U. Informática Ctra. Valencia Km. 7 E-28031 Madrid, España
[1] portiz@syst.upm.es,
[2]{jenifer.perez | salonso | jlsanchez | jgil}@eui.upm.es

1. Introducción

En la actualidad, la universidad española todavía tiene retos pendientes de superar para adaptarse al marco del Espacio Europeo de Educación Superior (EEES). Uno de los más importantes es el de proporcionar mecanismos para fomentar y evaluar la adquisición de competencias generales o transversales: capacidades, habilidades y/o aptitudes que el alumno debe desarrollar para aplicarlas a lo largo de su carrera profesional.Por otro lado, en la última década, las metodologías ágiles [1] han adquirido una gran relevancia en el área de la industria software, debido a que su adopción en las empresas está revirtiendo en un aumento de la calidad y competitividad en el mercado [2]. Esto hace indispensable que los contenidos necesarios para el aprendizaje de las metodologías ágiles se impartan en asignaturas pertenecientes a la rama de ingeniería del software, así como su práctica durante el ciclo formativo del ingeniero de software. Asimismo, las metodologías ágiles, y especialmente SCRUM [3], se centran en la gestión de proyectos basados en equipos auto-organizados donde se potencia a sus individuos. En este sentido, las metodologías ágiles se presentan como un marco perfecto para la adquisición de competencias generales de forma flexible y sencilla.

En este artículo se presenta la herramienta Agile Moodle, una plataforma de teleenseñanza especializada para soportar el Aprendizaje Ágil. El Aprendizaje Ágil es una solución para fomentar y evaluar las competencias generales al mismo tiempo que se aprende y se pone en práctica una de las metodologías ágiles más extendidas, SCRUM. El Aprendizaje Ágil consiste en adoptar los principios, valores, y prácticas de SCRUM en el ámbito educativo en general, y especialmente en el estudio de la Ingeniería del Software. Tanto el concepto de Aprendizaje Ágil, como su soporte mediante la herramienta Agile Moodle se enmarcan en el Proyecto de Innovación Educativa (PIE) Agile Learning [4].

La herramienta Agile Moodle sirve de guía para el desarrollo iterativo e incremental del aprendizaje de los futuros ingenieros de software ofreciéndoles utilidades para: (i) guiar y sustentar el proceso ágil SCRUM, (ii) el trabajo colaborativo en equipo mediante foros y wikis, uso de nuevas tecnologías de información y comunicación, (iii) aplicación de diferentes habilidades mediante asignación de roles, (iv) evaluación, co-evaluación y auto-evaluación de competencias específicas y generales mediante formularios, matrices cruzadas e informes; (v) portafolios en modo de estadísticas para el seguimiento del proceso de aprendizaje del alumno, evaluación continua, etc.

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 707-710, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

2. El contexto de aplicación

La Universidad Politécnica de Madrid y en concreto, la Escuela Universitaria de Informática [5], se encuentra inmersa de lleno en la implantación de los estudios de Grado correspondientes a los nuevos planes de estudio que se adaptan a los acuerdos firmados en Bolonia respecto del EEES.

En éste ámbito, el PIE *Agile Learning* [4], y por ende la herramienta *Agile Moodle*, está siendo aplicado de manera progresiva en aquellas asignaturas en las que se preveía una mejor adaptación a la metodología propuesta. Así, durante el curso 2010-2011 se empezó a aplicar en la asignatura optativa de *"Tecnologías de Desarrollo de Sistemas Web"* (TDSW), en la cual se desarrolló el programa de manera paralela en formato clásico y en el formato *agile*, de cara a poder comparar resultados. Se implantó también en la asignatura de *"Gestión de Proyectos y del RieSgo"* (GPRS), en la que se hizo sobre el total de los alumnos de la asignatura. Los resultados obtenidos en los informes de implantación de las distintas asignaturas han sido muy satisfactorios, especialmente en el grado de adquisición de competencias y en su evaluación, comparado con los resultados previos [6] a la implantación del PIE. Lo cual anima a seguir con la ampliación de su aplicación. Durante el presente curso 2011-2012 la aplicación se ha extendido a una nueva asignatura denominada *"Construcción Avanzada de Productos Software"* perteneciente al máster de investigación "Máster en Ciencias y Tecnologías de la Computación" y se mantiene en la asignatura GPRS, que en la actualidad están en pleno proceso de desarrollo.

3. Agile Moodle

Todo este soporte ágil para el aprendizaje y evaluación de asignaturas de ingeniería del software y por ende, aprendizaje de metodologías ágiles en un contexto académico, requiere de una gestión organizada a causa del volumen de equipos que se han de gestionar de forma simultánea. Para ello, se ha extendido la plataforma de enseñanza *Moodle*, añadiendo las funcionalidades requeridas para aplicar la metodología ágil SCRUM al aprendizaje. Esta extensión de la plataforma se conoce como *Agile Moodle* [4], y consiste en un conjunto de módulos desarrollados que se apoyan en funcionalidades ya ofrecidas por *Moodle* y además aportan otras nuevas. Concretamente, *Agile Moodle* ofrece nuevas funcionalidades para definir iteraciones, equipos, roles en equipos, y finalmente tanto la definición de reuniones retrospectivas y revisiones como su evaluación, dentro de lo que sería el curso normal de una asignatura.

SCRUM [3] es un método ágil para el desarrollo de proyectos mediante equipos que se auto-organizan en ciclos de corta duración que finalizan con una *revisión* del trabajo y una *retrospectiva*. Para planificar la asignatura en ciclos cortos, la herramienta Agile Moodle proporciona un denominado *Formato Agile Learning* (ver Fig. 1.a) que permite la división de un curso en iteraciones. Este formato muestra, además, los integrantes del grupo en todo momento. Asimismo, *Agile Moodle* está orientada al trabajo colaborativo basado en grupos de trabajo. A cada grupo se le ofrece un conjunto de herramientas colaborativas para facilitar el desarrollo de su trabajo. Estas

herramientas solo tienen visibilidad a nivel de grupo, lo que permite aislar los contenidos y asegurar la privacidad de los mismos. Además de esto, aporta la capacidad de establecer por cada iteración un responsable de grupo (*scrum master)*, además del responsable de apoyo del grupo y el propietario o (*Product Owner)*, cuyas figuras recaen en uno o más profesores (ver Fig. 1.b).

a) Definición Formato Iteración **b) Definición Grupo y Roles**

Fig. 1. Definición de una Asignatura Agile Learning

Fig. 2. Funcionalidades de Agile Moodle

Dentro de cada iteración se definen las tareas y trabajos a desarrollar, y se pone a disposición de los equipos el material y documentación necesario para adquirir los conocimientos específicos de la asignatura. El material que se imparte y el conjunto de entregables, conforman el *working product* de la iteración (ver Fig. 2). Durante cada iteración los alumnos se apoyan en la plataforma para consultar el material y realizar la entrega de las distintas tareas que se les ha asignado. Además pueden utilizar los recursos que ofrece la plataforma para el trabajo colaborativo, como son los foros y la mensajería instantánea. Mediante estos recursos se pueden realizar reuniones diarias/semanales, o se comparte documentos.

Al final de la iteración se celebran las reuniones de revisión y retrospectiva (ver Fig. 2), para evaluar las tareas realizadas y el grado de adquisición de competencias generales, respectivamente. Y para ello*, Agile Moodle* facilita dos actividades: retrospectiva y revisión, que permiten que los alumnos y el profesor puedan realizar la evaluación, auto-evaluación y co-evaluación del equipo durante esa iteración (ver Fig. 2). Estas evaluaciones se realizan mediante una tabla en la que se permite valorar del 1 al

10 cada una de las competencias generales, o las tareas, para cada uno de los miembros del equipo, obteniendo una nota de su auto-evaluación, co-evaluación, y evaluación del profesor de las competencias a nivel individual y en equipo (ver Fig. 3). Adicionalmente se obtienen informes que muestran la matriz de calificaciones de los alumnos y del grupo (ver Fig. 3). Finalmente, se proporciona un Portafolio que permite analizar y realizar un seguimiento de la evolución de los miembros del grupo durante el transcurso de la asignatura (ver Fig. 2).

Fig. 3. Evaluación y Generación de Informes de Retrospectivas

Agradecimientos

Este trabajo está cofinanciado por el PIE *Agile Learning* perteneciente a la E.U. Informática de la UPM y el proyecto *INNoSEP: INcorporing inNOvation in Software Engineering Processes* TIN2009-13849. Finalmente, los autores desean agradecer a Beatriz Blanco el trabajo realizado en el desarrollo de la plataforma Agile Moodle.

Referencias

1. Agile Manifesto, 2011, http://agilemanifesto.org/
2. Ambler S.: Agile Adoption Survey February 2008, http://www.agilemodeling.com/surveys
3. Schwaber K.: Agile Project Management with Scrum, Microsoft Press, 2004.
4. Plataforma Moodle Agile Learning, http://agilelearning.eui.upm.es/
5. Escuela Universitaria de Informática, http://www.eui.upm.es/
6. Díaz, J, Pérez J, Yagüe A, Alonso S, Gil J. 2011. Análisis Empírico del Papel de las Competencias Generales en el Marco de los Estudios Superiores. XVII Jornadas de Enseñanza de la Informática (JENUI 2011).

Auditoría de procesos de negocio en la nube: persistencia mediante almacenes no relacionales

M. Cruz, B. Bernárdez, M. Resinas, A. Durán

Departamento de Lenguajes y Sistemas Informáticos,
Universidad de Sevilla
{cruz, beat, resinas, amador}@us.es

Abstract. Cada día crece el número de aplicaciones y servicios *basados en la nube* ofertados por proveedores tales como Amazon, Google o Sun entre otros. Además de ofrecerse el software como servicio (*SaaS, Software as a Service*), destacan los sistemas de procesos de negocio que se ofrecen a los clientes como servicio, denominados PRaaS (*Process as a Service*).

Uno de los problemas que conlleva la computación en la nube (*cloud computing*) es la pérdida de control sobre los datos y procesos que hace necesario la realización de auditorías que permitan además verificar el grado de cumplimiento de los procedimientos y regulaciones establecidas por la organización. Como resultado de este proceso de auditoría, es habitual que el volumen de datos se incremente considerablemente en poco tiempo por lo que la escalabilidad será uno de los requisitos a exigir al sistema de almacenamiento subyacente.

En este trabajo se plantea una posible línea de investigación basada en el uso de sistemas de bases de datos no relacionales para dotar de persistencia a los sistemas PRaaS persiguiendo el principal objetivo de mejorar la escalabilidad de los mismos.

Keywords: PRaaS, bases de datos no relacionales, cloud computing, compliance management, escalabilidad

1 Introducción

Actualmente está aumentando el número de organizaciones que aprovechan la tecnología de computación en la nube. Esta tecnología permite utilizar únicamente los recursos necesarios en cada momento lo que produce una reducción de los costes ya que únicamente se paga por los recursos que se utilizan.

Los servicios y aplicaciones ofrecidos para su contratación mediante computación en la nube son variados, por ejemplo, software como servicio, infraestructura como servicio, etc. Actualmente además las empresas pueden contratar la gestión de los procesos de negocio, lo que se conoce como PRaaS (*Process as a Service*).

Uno de los problemas de la computación en la nube es la pérdida de control que se produce sobre los datos y ejecuciones [7]. Centrándonos en el PRaaS ese control es fundamental cuando las organizaciones se someten a auditorías de la conformidad de

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 711-716, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

los procesos desplegados en su sistema [1]. El propósito de estas auditorías de conformidad es determinar si el auditado actúa, o ha actuado, de acuerdo a los procedimientos y regulaciones establecidas por una autoridad, por ejemplo: leyes (del tipo de Sarbanes-Oxley en EEUU [21], que regula las funciones financieras contables y de auditoría, o la LOPD en España), buenas prácticas, o recomendaciones (como ITIL [22] o EFQM [10]) [3].

El resto del artículo se organiza de la siguiente manera. En la sección 2 se identifican los principales problemas de la conformidad en los PRaaS. En la sección 3 se introducen algunos conceptos y en la sección 4 se describe el enfoque de la investigación que se pretende abordar.

2 Principales aspectos por resolver en la conformidad de los PRaaS

Aunque existe una importante cantidad de trabajo realizado en el estudio de la conformidad [2, 9,12], aún queda trabajo por hacer al respecto:

Por un lado, la mayoría de las aproximaciones para la comprobación de la conformidad se centran en un tipo concreto de regla de conformidad, por ejemplo las relacionadas con flujo de control, y con un momento concreto de comprobación de la conformidad: antes [2, 8], durante [4] y después [11] de la ejecución de un proceso de negocio. Sin embargo, hasta donde sabemos, no se ha definido aún un sistema de gestión de la conformidad que de soporte al ciclo de vida completo de los procesos de negocio [6].

Por otro, para hacer posible las auditorías de conformidad es necesario dejar un registro de cómo transcurren los procesos de negocio generando evidencias que permitan verificar la conformidad posteriormente. Las evidencias son habitualmente recogidas en forma de *logs* de eventos y de datos manejados y obtenidos como resultado de las distintas actividades del proceso de negocio.

Hasta el momento para dotar de persistencia a los datos generados, lo más habitual es utilizar sistemas de gestión de bases de datos relacionales para hacer frente a una gran cantidad de datos generados [5]. Los sistemas de procesos de computación en la nube deben soportar un gran número de usuarios y alta carga de transacciones lo que puede provocar que haya muchas instancias con un gran volumen de datos ejecutándose simultáneamente que hacen necesaria la escalabilidad.

Los sistemas relacionales presentan el problema de no escalar fácilmente por lo que al aumentar la carga de los mismos el rendimiento puede disminuir de forma exponencial. Habría que estudiar la manera más adecuada para almacenar los datos persiguiendo una implementación fácilmente escalable.

Desde hace unos años han surgido los sistemas NoSQL [20] como un nuevo paradigma en la gestión de datos para dar respuesta a las nuevas necesidades de almacenamiento de datos para sistemas web, redes sociales, juegos online, computación en la nube, etc. Dado que una de sus principales características es la escalabilidad parece que se podrían adaptar bien a los sistemas de conformidad de los PRaaS.

Existen ya trabajos [23] en los que se aplican bases de datos NoSQL para mejorar la escalabilidad en sistemas que utilizaban sistemas relacionales. En dicho trabajo se propone una solución de persistencia mediante una base de datos NoSQL orientada a documento para sistemas de desarrollo software basado en modelos (MDE).

3 Background

En esta sección se definen los principales conceptos necesarios para comenzar a abordar la investigación y se resumen las propiedades de las bases de datos NoSQL.

3.1 Escalabilidad

La escalabilidad es la capacidad de un sistema de atender un número creciente de peticiones y seguir cumpliendo con los requisitos impuestos. La forma de conseguir la escalabilidad puede ser vertical u horizontal [18]. Para conseguir escalabilidad vertical es necesario añadir recursos tales como CPU, memoria, etc. al servidor pero el coste va aumentando de forma exponencial y llega un momento en el que ya no puede seguir escalando. La escalabilidad horizontal consiste en añadir más servidores en paralelo para distribuir los datos. Algunos sistemas proporcionan a la vez escalabilidad horizontal y vertical.

Los sistemas de bases de datos relacionales almacenan los datos en tablas interrelacionadas; para las consultas necesitan combinar tablas que pueden estar en distintos servidores [19]. Los sistemas de NoSQL no presentan este problema ya que los datos no se almacenan normalizados y pueden estar distribuidos en varios servidores según el valor de la clave. Estos sistemas escalan con facilidad en forma horizontal.

3.2 Propiedades ACID

Otra diferencia importante entre los sistemas relacionales y los sistemas de almacenamiento NoSQL es relativa al cumplimiento de las propiedades ACID.

Las BD relacionales soportan el concepto ACID [19]. Sus transacciones tienen que cumplir los requisitos de **A**tomicidad, **C**onsistencia, a**I**slamiento y **D**urabilidad.

Sin embargo, los almacenamientos de datos NoSQL se caracterizan por su escalabilidad y disponibilidad. Estos dos objetivos se consiguen mediante partición (datos distribuidos en varios servidores para facilitar la escalabilidad) y replicación (datos duplicación en varios nodos o servidores para conseguir que el sistema esté siempre disponible)[15].

Como consecuencia de las necesidades de escalabilidad y disponibilidad se hace necesario relajar las condiciones ACID [18].

Eric Brewer presenta en el año 2000 el teorema CAP [14, 13], en el que se propone que para un sistema distribuido, sólo dos de las siguientes tres propiedades pueden ser satisfechas a la vez:

- Consistencia *(**C**onsistence)*. Cuando un dato esté replicado en varios nodos, su valor debe ser el mismo en todos los nodos en que esté replicado. Es equivalente a la C de ACID. Afectará a la disponibilidad.

- Disponibilidad *(**A**vailability)*. El sistema siempre es capaz de responder. Si se aumenta la disponibilidad disminuye la consistencia.
- Tolerancia a fallos *(**P**artition-tolerance)*. El sistema debe seguir funcionando aunque se pierdan de forma arbitraria algunos mensajes debido a fallos de alguna parte del sistema.

Cuando se produce una actualización de datos, para conseguir que haya consistencia, será necesaria su propagación al resto de las réplicas, por lo que la disponibilidad puede verse afectada temporalmente. En muchas de las implementaciones de NoSQL lo que se abandona es la consistencia como tal, garantizando que con el tiempo la base de datos sí será consistente *(eventually consistent)*. Implica que con el tiempo los datos estarán actualizados en todos los nodos [17].

Aunque la mayoría de las aplicaciones necesitan transacciones que cumplan ACID, hay sistemas como por ejemplo redes sociales en las que no es crítico que algún mensaje no se visualice en forma instantánea.

3.3 Características de los sistemas NoSQL

Una posible clasificación de los sistemas NoSQL junto con algunas de sus características se muestra en la tabla 1.

Tipo de BD NoSQL	Características de Almacenamiento	Principales ventajas	Implementaciones
BD clave-valor (*Key-value*)	Se almacenan parejas clave: valor; valores identificados por una clave.	Muy rápido para lecturas.	Dynamo (Amazon) [15], Voldemort, Tokyo Cabinet
BD de columnas (*Column store*)	Almacena los datos en columnas en lugar de filas.	Consultas de datos por columnas y agregados.	BigTable (Google) [16], Cassandra (Facebook)y HBase
BD orientadas a documentos (*Document Store*)	Similar al clave-valor pero el valor es un documento que la BD puede interpretar.	Consultas complejas sobre campos del documento que se almacena.	Apache CouchDB, MongoDB
BD orientadas a grafos (*Graph databases*)	La información se guarda en forma de grafo con nodos, arcos (conexiones) y propiedades de arcos y nodos.	Consultas sobre nodos y conexiones.	Neo4j

Tabla 1. Clasificación de las principales bases de datos NoSQL

Estas bases de datos se caracterizan por su escalabilidad y disponibilidad por lo que responden bien en situaciones de gran volumen de datos, no tienen esquema fijo de datos y la mayoría de las implementaciones son de código abierto [19].

4 Hipótesis de investigación y plan de trabajo

Tras identificar los problemas de escalabilidad de la gestión de evidencias en PRaaS, la hipótesis de partida de la investigación que se pretende someter a estudio es "Puede ser beneficioso usar una base de datos NoSQL para mejorar la escalabilidad de sistemas de gestión de evidencias en procesos que se ejecutan en la nube".

El objetivo que se pretende alcanzar es probar que al usar una base de datos NoSQL se obtienen mejoras respecto a las bases de datos relacionales en relación a variables como escalabilidad y disponibilidad.

Para comenzar la investigación una posibilidad es plantear un caso de estudio consistente en las siguientes tareas:

1. Definir una batería de procesos de negocio para su gestión en la nube.
2. Definir indicadores que permitan medir la escalabilidad, disponibilidad y otros parámetros relativos al rendimiento que se identificarán durante la investigación.
3. Definir una primera aproximación de conformidad o normas a cumplir por los procesos de negocio desplegados en la nube.
4. Registrar las evidencias que se generan durante la instanciación del proceso de negocio. La idea esencial es desplegar los procesos de negocio en una base de datos NoSQL y en un sistema de bases de datos relacional.
5. Medir y comparar el comportamiento de ambas bases de datos respecto a las variables identificadas.

Se espera que los resultados del caso de estudio proporcionen realimentación para seguir avanzando. En concreto para poder estudiar qué base de datos NoSQL es más conveniente según las características de los procesos de negocio o la naturaleza de las reglas de conformidad fijadas por la auditoría.

Referencias

1. R. Accorsi. Business Process as a Service. Chances for Remote Auditing. IEEE 35th Annual Computer Software and Applications Conference Workshops, pag. 398–403. IEEE-2011.
2. A. Awad, G. Decker, and M. Weske. Efficient compliance checking using bpmn-q and temporal logic. In BPM, pag. 326–341, 2008.
3. J. Bace and C. Rozwell. Understanding the components of compliance. Gartner Research Paper, 2006.
4. A. Birukou, V. D'Andrea, F. Leymann, J. Serafinski, P. Silveira, S. Strauch, and M. Tluczek. An integrated solution for runtime compliance governance in soa. In ICSOC, pag. 122–136, 2010.
5. J. Dean and S. Ghemawat. MapReduce. Communications of the ACM, 51(1):107, 2008.
6. C. Cabanillas, M. Resinas, and A. R. Cortés. Exploring features of a full-coverage integrated solution for business process compliance. In C. Salinesi and O. Pastor, editors, CAiSE Workshops, volume 83 of Lecture Notes in Business Information Processing, pag. 218–227. Springer, 2011.

7. R. Chow, P. Golle, M. Jakobsson, E. Shi, J. Staddon, R. Masuoka, and J. Molina. Controlling data in the cloud: outsourcing computation without outsourcing control. In Proceedings of the 2009 ACM workshop on Cloud computing security, pages 85–90. ACM, 2009.
8. Y. Liu, S. Müller, and K. Xu. A static compliance-checking framework for business process models. IBM Systems Journal, 46(2):335–362, 2007.
9. K. Namiri and N. Stojanovic. Using control patterns in business processes compliance. InWeb Information Systems Engineering - WISE 2007Workshops, pag. 178–190, 2007.
10. S. Rinderle-Ma, L. T. Ly, and P. Dadam. Business process compliance. EMISA Forum, 28(2):24–29, 2008.
11. A. Rozinat and W. M. P. van der Aalst. Conformance checking of processes based on monitoring real behavior. Information Systems, 33(1):64–95, 2008.
12. S. Sadiq, G. Governatori, and K. Namiri. Modeling control objectives for business process compliance. In Business Process Management, Volume 4714, pag. 149–164, 2007.
13. S. Gilbert, N. Lynch. Brewer's. Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services. ACM SIGACT News, 33(2):51-59, 2002.
14. E. Brewer. Towards Robust Distributed Systems, Proceedings of Principles of Distributed Computing (PODC 2000), 2000.
15. G. DeCandia y otros. Dynamo. Amazon's Highly Available Key-value Store. Proceedings of symposium on Operating systems principles (SOSP), pag. 205-220, 2007.
16. F. Chang y otros. Bigtable. A Distributed Storage System for Structured Data. Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation, 2006.
17. http://www.allthingsdistributed.com/2008/12/eventually_consistent.html, 2012.
18. R. Cattell. Scalable SQL and NoSQL Data Stores. SIGMOD Rec. 39(4):12-27, 2010.
19. N. Leavitt. Will NoSQL Databases Live Up to Their Promise?. IEEE Computer Society 43(2):12-14, 2010.
20. D. McCreary. The CIO's Guide to NoSQL, 2011. https://s3.amazonaws.com/dataversity/documents/CIO_Guide_NoSQL_v4.pdf, 2012.
21. S. Bainbridge. Complete Guide to Sarbanes-Oxley : Understanding How Sarbanes-Oxley Affects Your Business, 2007.
22. L. Klosterboer. ITIL Capacity Management. IBM Press, 2011.
23. J. Espinazo, J. Sánchez, J. García. Morsa: A Scalable Approach for Persisting and Accessing Large Model http://www.springerlink.com/content/8264815854324251/fulltext.pdf

Sistedes 2012
#sistedes2012 Almería

JISBD PROLE JCIS

ACG
Applied Computing Group
Ref. TIC-211, Junta de Andalucía
University of Almería, Spain
Ctra. Sacramento s/n, 04120 Almería

FUNDACIÓN MEDITERRÁNEA
EMPRESA - UNIVERSIDAD DE ALMERIA

Colaboradores

GOBIERNO DE ESPAÑA
MINISTERIO DE ECONOMÍA Y COMPETITIVIDAD

JUNTA DE ANDALUCIA
CONSEJERÍA DE ECONOMÍA, INNOVACIÓN Y CIENCIA

DIPUTACIÓN DE ALMERÍA

AYUNTAMIENTO DE ALMERÍA

ALBORADA Ingeniería I+D+I

Escuela Superior de Ingeniería
UNIVERSIDAD DE ALMERÍA

UNIVERSIDAD DE ALMERÍA
Vic. de Tecnologías de la Información y de la Comunicación
Vic. de Investigación, Desarrollo e Innovación
Unidad de Enseñanza Virtual del VTIC (EVA/TIC)
Departamento de Lenguajes y Computación

UNIVERSIDAD DE ALMERÍA
Plan Propio de Investigación

cajamar

PITA Parque Científico-Tecnológico de Almería

IEEE computer society
IEEE Computer Society – Spanish Chapter

ices Ingeniería, Consultoría y Estrategia de Sistemas

almerimatik i+d+i

http://sistedes2012.ual.es