Una metodología para el desarrollo de software basado en COTS

Luis Iribarne[1] y Antonio Vallecillo[2]

[1]Dpto. Lenguajes y Computación
Universidad de Almería
liribarn@ual.es

[2]Dpto. Lenguajes y Ciencias de la Computación
Universidad de Málaga
av@lcc.uma.es

Resumen

Uno de los objetivos tradicionales de la ingeniería del software ha sido la necesidad de desarrollar sistemas software mediante el ensamblaje de módulos independientes software. En este sentido, los componentes software reutilizables han jugado un papel importante. Por otro lado, están apareciendo un gran número de trabajos que hacen uso de componentes comerciales —también llamados COTS— en las fases de desarrollo de un sistema software. Uno de los problemas que presenta este tipo de componente es que, por regla general, carece de información suficiente para que puedan ser tratados en procesos automáticos de búsqueda de componentes desde repositorios, para su composición respetando la arquitectura de la aplicación software. Además, tampoco existen metodologías conocidas que ayuden en estas tareas de composición. En este trabajo se presenta una metodología a nivel global para el desarrollo de sistemas software basados en COTS, y unos objetivos generales que extienden dicha metodología. Las ideas propuestas en este trabajo pretenden servir como base de un trabajo de investigación en el campo de la ingeniería del software basada en componentes COTS.

1 Motivación y plantemientos generales

Una de las disciplinas de la ingeniería del software que más impulso está teniendo en los últimos tiempos es aquella que describe, desarrolla y utiliza técnicas basadas en componentes software para la construcción de sistemas abiertos y distribuidos, como por ejemplo los sistemas de información distribuidos bajo Web. Esta disciplina también se conoce como ingeniería del software basada en componentes distribuidos.

Los componentes software en cierta medida surgen ante la necesidad de hacer un uso correcto de software reutilizable dentro de las aplicaciones. Sin embargo, la mayoría de las metodologías de reutilización actuales no contemplan el uso de componentes comerciales ya existentes para el desarrollo de aplicaciones software. A este tipo de software se le conoce con el nombre de *"commercial-off-the-shelf"* o componente COTS. Con el término COTS queremos hacer referencia a aquel software comercial desarrollado previamente por terceras partes, fuera de las estrategias de desarrollo y aplicadas durante todo el ciclo de vida del producto a desarrollar en base a productos comerciales, entre los que se incluyen: los sistemas heredados (*legacy systems*), software de dominio público, y otros elementos desarrollados fuera de la organización, también llamados software NDI (*Non-Developmental Item*) [2]. Esta clase de componente software generalmente es adquirido en formato binario, sin posibilidad de tener acceso al código fuente y sin información adicional que ayude a los integradores en la selección correcta de los mismos. Ante este marco, pensar en tareas para la automatización de los procesos es prácticamente imposible:

(a) ¿Cómo se lleva a cabo actualmente la localización, selección e integración de componentes COTS?

(b) ¿Existen repositorios de componentes COTS y traders adecuados?

(c) ¿Qué tipo de información ofrece un vendedor COTS a sus clientes?

(d) ¿Qué tipo de información es la que debe utilizar un cliente para que el vendedor sepa lo que quiere?

(e) ¿Cómo aunar la información de los clientes y de los vendedores para poder seleccionar componentes, asegurando que la aplicación construida con ellos satisface los requisitos de las especificaciones?

El problema consiste en determinar el flujo de información que existe, por un lado, desde el vendedor hacia el cliente, que intenta decir "esto es lo que tengo, dime lo que quieres" y por otro lado desde el cliente hacia el vendedor, que intenta decir "esto es lo que quiero, dime que lo que tienes". El verdadero problema es que tal flujo de información no existe. Es decir, el vendedor COTS no vende su producto, no ofrece información suficiente a sus clientes para que estos puedan determinar, en primer lugar, si los componentes que les ofrece el vendedor son válidos para sus arquitecturas diseñadas, y en segundo lugar, cómo integrar los componentes en dichas arquitecturas a partir de la información disponible. Por otro lado, el cliente sabe lo que necesita pero no sabe cómo hacérselo llegar a su/s vendedor/es. Por muchos intentos por ambas partes en recopilar el máximo de información posible, en la mayoría de los casos el vendedor desconocerá detalles técnicos que su cliente omite en sus preguntas por considerarlos obvios, y en mayor grado sucede a la inversa.

Parte de estos problemas se podrían resolver si existiera un mecanismo, una forma o una técnica unificada para la especificación de componentes COTS tanto para los vendedores como para los clientes que buscan un determinado producto. Como resultado final en la elaboración de su producto, los vendedores o encargados de desarrollar componentes COTS, pueden recoger o especificar sus componentes a partir de unas plantillas que los clientes entenderán, ya que estos utilizarán unas plantillas similares para el caso inverso: solicitar un producto. Incluso estas especificaciones basadas en plantillas podrían servir como fuente de información para posibles repositorios de componentes COTS que los traders o brokers, encargados de su localización, podrían utilizar para automatizar los procesos descritos anteriormente. Por tanto, es de interés definir aquellos elementos y procesos que conduzcan a una metodología para el desarrollo de aplicaciones software basadas en componentes COTS. En lo que resta de este trabajo intentamos justificar la necesidad de esta metodología y describimos los pasos que vamos a seguir para alcanzar este objetivo.

Este trabajo está organizado en cuatro apartados, siendo el primero de ellos el que concierne a los planteamientos y motivaciones globales descritos hasta el momento. En segundo lugar, en la sección 2 se establece el marco de trabajo donde intervienen los tres elementos básicos de la metodología: la arquitectura software, los repositorios y el proceso de intermediación, también conocido como trader. Seguidamente, en la sección 3 se exponen cuáles son los objetivos concretos que pretendemos alcanzar con esta metodología basada en COTS. Finalizamos el presente trabajo con unas valoraciones generales y conclusiones en la sección 4.

2 Una metodología basada en COTS

Como hemos adelantado antes, el marco de trabajo en el cual nos vamos a centrar contempla la necesidad de establecer una metodología para el desarrollo de aplicaciones distribuidas basadas en componentes COTS. Desde una perspectiva global, como la que se muestra en la figura 1, esta metodología puede quedar identificada por la tríada (A,T,R): una arquitectura, un trader y un repositorio.

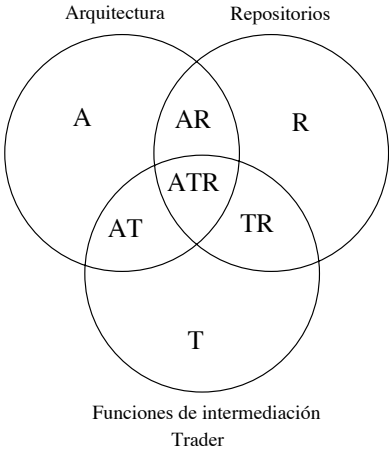

Figura 1: Visión ATR de la metodología basada en COTS

Según la visión ATR anterior, en primer lugar está la arquitectura de la aplicación software (A), desde donde se definen las características de los componentes que intervendrán en la aplicación, como sus interfaces, su protocolos de interacción y/o sus nexos de unión. En segundo lugar está el repositorio (R) donde se almacena la información de los componentes existentes. Actualmente existen dos clases de respositorios, los repositorios IDL y los repositorios de implementación. En tercer lugar está el proceso de intermediación (T), conocido como Trader. Este proceso será el que se encargue de comprobar qué componentes del repositorio cumplen las restricciones impuestas en la especificación de la arquitectura software. Según esto, hay que destacar dos tipos de especificaciones en este marco: por un lado aquellas especificaciones concretas de los componentes software que residen en los repositorios, y por otro lado las especificaciones abstractas de los componentess definidas en las arquitecturas software [4]. El proceso Trader por tanto, utilizará estas especificaciones en sus actividades de búsqueda y selección.

Como se puede comprobar en la figura 1, existen partes comunes en las tres entidades ATR que sirven de vías de interoperabilidad entre ellas —AT, TR, AR y ATR—. Pero debido a la falta de información disponible en un componente COTS —acceso al código fuente, especificaciones de comportamiento, casos de uso y restricciones— y también debido a la falta de una metodología de desarrollo concreta para esta clase de componente, son precisamente estas áreas de interconexión del ATR las que se presentan como lagunas por resolver para un desarrollo basado en COTS. Por todas estas razones, pensamos que son necesarias definir unas plantillas de especificación de componentes comunes para AT y TR con el propósito de lograr la conexión A-R entre los componentes especificados en la arquitectura A y los componentes implementados residentes en el repositorio R.

2.1 Los repositorios

Los repositorios actuales almacenan información de las interfaces en forma de IDLs, lo cual permitiría a un proceso trader llevar a cabo actividades de compatibilidad sintáctica. Sin embargo, siguiendo la línea de los trabajos existentes [6, 7], es necesario extender estos repositorios para que contemplen información semántica y de protocolos de los componentes —esta extensión debería quedar también reflejada en las plantillas de especificación—. Según esto, ahora es necesario definir cuatro tipos de repositorios: un repositorio de implementación, un repositorio IDL, un repositorio de protocolos y un repositorio semántico. Esta extensión podría ser tratada mediante MOF (Meta-Object Facility) [3], una utilidad de OMG (Object Management Group) para la gestión de metadatos en sistemas distribuidos basados en CORBA. Además, la especificación MOF cumple con otras especificaciones de OMG también muy conocidas, como por ejemplo UML, XMI o CCM.

2.2 La arquitectura software

Por otro lado, la metodología para el desarrollo de software basada en COTS que pretendemos elaborar deberá trabajar con las especificaciones formales de arquitecturas software, en la línea de los llamados lenguajes de definición de arquitecturas (LDAs). En la literatura existe una gran variedad de estos lenguajes, y entre ellos destacamos por ejemplo ACME, AESOP, Wright (estos tres de Carnegie Mellon University), Rapide, Darwin o LEDA, entre otros.

2.3 La función de intermediación (el trader)

Por último, el objetivo de una función de intermentación o proceso trader, es proporcionar el medio de ofrecer y descubrir instancias de un determinado tipo de servicio con determinadas características [5]. En este proceso de intermediación intervienen diversos miembros u objetos con diferentes roles (cometidos), actividades y políticas de intermediación. Por ejemplo, existe un rol que se denomina *intermediario* que registra ofertas de servicio de los objetos exportadores y devuelve ofertas de servicio a los objetos importadores. También existen los roles *exportador* e *importador*, que registran ofertas de servicio con el objeto intermediario, y que obtienen ofertas de servicio del objeto intermediario bajo ciertos criterios, respectivamente. En cuanto a las actividades se refiere, están la actividad de exportación de servicios y la actividad de importación de servicios. En este sentido, es necesario extender estas funciones o actividades de intermediación del proceso trader para que refleje el uso de las plantillas de especificación y las clases de repositorios comentados anteriormente.

3 Objetivos concretos

En este apartado enumeramos los objetivos que pretendemos alcanzar para la elaboración de la metodología basada en COTS presentada. Estos objetivos generales se pueden resumir como:

(a) Extender la especificación de un componente para que cubra información sintáctica, semántica y de protocolos, elaborando unas plantillas de especificación al estilo [1]. Se puede hacer uso de notaciones formales como Object-Z, Larch o álgebras de procesos —entre otros— con la posibilidad de tratar de combinar algunos formalismos.

(b) Extender los repositorios actuales para que puedan almacenar información sintáctica, semántica y de protocolos. Se puede hacer uso de implementaciones basadas en dMOF de DSTC [3].

(c) Extender la especificación de las funciones de intermediación en las actividades de oferta y demanda de servicios para que vengan condicionadas por la información de las plantillas de especificación o por restricciones impuestas por el desarrollador del sistema.

(d) Extender las estrategias de búsqueda y selección de componentes COTS que contemplen múltiples componentes, y componentes con múltiples interfaces.

(e) Extender los traders actuales para que hagan uso de las estrategias del apartado (d).

(f) Establecer unas métricas y heurísticas que nos permitan evaluar los resultados del trader y que ayuden en las tareas de toma de decisiones respecto a su grado de cumplimiento con los componentes definidos en la arquitectura software.

(g) Describir, desarrollar y formalizar una metodología para el desarrollo software basado en componentes COTS que contemple las extensiones anteriormente citadas.

(h) Elaboración de un prototipo CORBA a nivel de protocolos basado en un LDA concreto. El prototipo preferiblemente deberá estar implementado en Java, y podrá utilizar cualquier ORB de libre distribución, como ORBacus, JavaORB o JacORB. La plataforma seleccionada será Linux/RedHat.

4 Conclusiones

En estos últimos años se ha podido comprobar un aumento en la utilización de componentes comerciales COTS en las fases de desarrollo de los sistemas basados en componentes software. Sin embargo, no existe ninguna metodología de desarrollo para estos entornos ni especificaciones formales que contemplen información de componentes COTS. En este trabajo se ofrece una visión global de una metodología para el desarrollo basado en COTS. En esta visión intervienen tres elementos básicos. En primer lugar, una arquitectura software, donde se definen los requisitos que deben tener los componentes de la aplicación. En segundo lugar, un conjunto de repositorios, donde residen los IDLs y las implementaciones, junto con información semántica y de protocolos. En último lugar, unas funciones de intermediación para la gestión de actividades de oferta y demanda de servicios entre la arquitectura y los repositorios. Para finalizar, en este trabajo también se presentan unos objetivos concretos que pretendemos alcanzar como investigación dentro del campo de la ingeniería del software basada en componentes COTS distribuidos.

Referencias

[1] E. A. Addy and M. Sitaraman. Formal specification of COTS-based software: A case study. In *Proc. Fifth Symposium on Software Reusability*, pages 83–91, may 1999.

[2] D. Carney and F. Long. What do you mean by COTS? Finally, a usefull answer. *IEEE Sof*,17(2):83–86, 2000.

[3] DSTC. *dMOF User Guide*. Distributed Systems Technology Centre, Jul. 2000.

[4] L. Iribarne and A. Vallecillo. Sobre la búsqueda y emparejamiento de componentes COTS con múltiples interfaces. In *JISBD'2000*. Jornadas de Ingeniería del Software y Bases de Datos, Nov. 2000.

[5] ISO/IEC-ITU/T. Trading function: Specification. ISO/IEC 13235-1, ITU/T X.950, 1998.

[6] G. T. Leavens and M. Sitaraman. *Foundations of Component-Based Systems*. Cambridge Univ. Press, 2000.

[7] A. Vallecillo, J. Hernández, and J. M. Troya. Object interoperability. In *Object-Oriented Technology: ECOOP'99 Workshop Reader*, number 1743 in LNCS, pages 1–21. Springer-Verlag, 1999.