Using COTS-Widgets Architectures for Describing User Interfaces of Web-Based Information Systems

Article *in* International Journal of Knowledge Society Research · July 2011

DOI: 10.4018/ijksr.2011070106 · Source: DBLP

CITATIONS
8

READS
319

4 authors, including:

Luis Iribarne
Universidad de Almería
163 PUBLICATIONS **1,235** CITATIONS
SEE PROFILE

Javier Criado
Universidad de Almería
72 PUBLICATIONS **475** CITATIONS
SEE PROFILE

Nicolás Padilla
Universidad de Almería
52 PUBLICATIONS **309** CITATIONS
SEE PROFILE

L. Iribarne, J. Criado, J.A. Asensio, N. Padilla.(2011)Using COTS-Widgets Architectures for Describing User Interfaces of Web-Based Information Systems. International Journal of Knowledge Society Research (IJKSR) 2(3):61-72. IGI-Global. ISSN: 1947-8429

Using COTS-*Widgets* Architectures for Describing User Interfaces of Web-Based Information Systems

Luis Iribarne, Applied Computing Group, University of Almeria, Spain

Javier Criado, Applied Computing Group, University of Almeria, Spain

Nicolás Padilla, Applied Computing Group, University of Almeria, Spain

José-Andrés Asensio, Applied Computing Group, University of Almeria, Spain

ABSTRACT

Modern Web-based Information Systems (WIS) must be flexible and prepared to be easily accessible and manageable in real-time. WIS user interfaces (UI) are still being constructed on the basis of traditional software development paradigms, without taking into account in their construction (or in the knowledge managed by the systems) the main criterion of globalization, that they must be distributed, open and changing. WIS-UI must be able to be constructed depending on the type of interaction (individual or collective) and the purpose of the interaction (management, technical, etc.). In this paper, the authors present a component-based development perspective to build user interfaces of WIS by means of the composition of widgets-components architectures and MDD approaches.

Keywords: *COTS Components, Cotsgets, Model-Driven Development, User Interfaces, Web-based Information Systems (WIS), Widgets*

1. INTRODUCTION

Modern *Web-based Information Systems* (WIS) must be flexible, adaptable, extensible, accessible and manageable by different persons and/or groups of persons with common interests located in different places. Recently, special interest has been given to globalization of information through a common system vocabulary using ontologies and web semantics. A great effort has also been devoted to recalling information on the Web, with powerful search engines based on ontologies and intelligent software agents, a mechanism known in the literature as "*information retrieval*". However, at present, WIS user interfaces (UI) are still being constructed on the basis of traditional software development paradigms, without taking into account in their

http://doi.org/10.4018/ijksr.2011070106

construction (or in the knowledge managed by the systems) the main criterion of globalization, that they must be distributed, open and changing. This means that a WIS UI must be able to be dynamically reconstructed depending on the type of interaction (individual or collective) and the purpose of the interaction (management, technical, etc.).

WIS modernization techniques are aimed at the same priority consideration as the ICT EU of 7th Framework Programme: "The future of the Internet" (around the new Web 3.0). Some techniques of our interest are: (a) Web information retrieval applied to UI mediation services; (b) Semantic Web for cooperative UI intelligence; (c) Decision mechanisms; (d) Composition architectures in real time: for UI composition; (e) Multi-agent architectures: interface agents, mediation agents, etc.

Within this framework, our interest has been focussed on studying and developing an experimental methodology to work with WIMP-type simple user interfaces *(Windows, Icons, Menus and Pointers)* (Almendros & Iribarne, 2008). Such user interfaces are based on "bottom-up" composition of *widgets*-type COTS interface components (that we called *cotsgets*). The methodology is inspired on basic principles of *Model-Driven Development* (MDD) (Asadi & Ramsin, 2008; Stahl & Völter, 2006). A WIS/WIMP-*costgets* user interface is considered as an architecture based on *cotsgets*-type components; such architecture respects some principles of composition like dependence between components, restrictions in use, availability and visibility, etc. As we focussed on an MDD-based solution, here we'll consider an *architecture* as a WIS/WIMP-*cotsgets* interface meta-model. The meta-model instatiation, with the specific *cotsgets*-type components, represents the UI made up of "portions".

A WIS example requiring a solution for this situation gap are *Environmental Management Information Systems* (EMIS) (International Organization for Standardization, 1996). EMIS are social and technical systems with a variety of final users and actors (i.e., politicians, technicians, administrators, etc.) that cooperate with each other and interact with the system by means of powerful and strict UI for decision-making, problems resolution, etc.

SOLERES is a Web-based EMIS which sets up a framework for correlating satellite maps and ecological cartography using neural-networks (http://www.ual.es/acg/soleres). This information system, like other current WIS, must be flexible and allow simple, quick access to promote globalization of the information. To accomplish this goal, our environmental information system was basically designed in two large subsystems. On one hand, all of the infrastructure and platform supporting the information system knowledge base (SOLERES-KRS), and on the other, all human-computer interaction (SOLERES-HCI). The system for representing knowledge, implemented in SOLERES-KRS, was modeled using ontologies, which allows the system to have meta-information repositories of satellite images and cartography, with the original information in repositories outside of the system. The platform was prepared to incorporate information from correlating satellite images and cartography. This subsystem, and also the system for representing the knowledge that it implements, has been widely described elsewhere (Asensio et al., in press; Iribarne et al., 2010; Padilla et al., 2008). On the other hand, the SOLERES-HCI subsystem manages exploitation of the information (in this case environmental), facilitating interaction with user interfaces that mediate for the users in searching for and exploiting the information and facilitating decision-making tasks (environmental), and prediction/prevention. Our proposal for building WIS/EMIS user interfaces is a real-time approach inspired on composition perspective with COTS-interface components (type interface *widgets*).

This article deals with the SOLERES-HCI subsystem. In section 2 we'll briefly describe the proposal's state of the art. In Section 3, we'll offer an example scenario to explain the proposal. Then, in Section 4, we'll show a COTS-based MDD perspective for automatic composition of user interfaces. We'll end up with some conclusions and future work in Section 5.

http://doi.org/10.4018/ijksr.2011070106

2. RELATED WORK

This section briefly reviews areas related to the main parts of this work: (a) EMIS works; (b) UI modeling and construction following COTS and MDD perspectives.

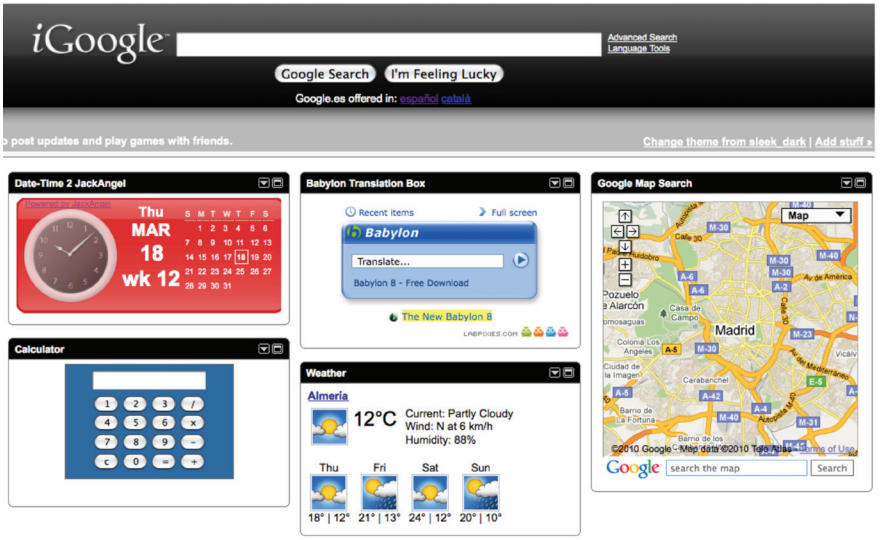
An EMIS is a special kind of *Geographic Information System* (GIS) (International Organization for Standardization, 1996). There are several different kinds of EMIS in the literature, which use some of the technologies in our system. For instance, InfoSleuth (http://www.research.telcordia.com/InfoSleuth/) is a distributed EMIS based on agents, able to offer complex consulting services on heterogeneous resources, and is based on ontologies, services or interaction templates. EDEN-IW (http://www.eden-iw.org) is an EMIS, which allows information recall, treatment and location with an intelligent UI and support tools depending on the access profile. NZDIS (Cranefield & Purvis, 2001) is an EMIS based on a multi-agent system describing the architecture for constructing distributed information systems based on existing sources of heterogeneous information. FSEP (Dance et al., 2003) is a weather prediction EMIS based on agents and components for distributed execution. None of these systems use COTS components for building user interfaces.

There are many model-based tools for generating UIs in the literature: IDEAS (*Interface Development Environment within OASIS*) (Lozano et al., 2000), a model-based UI development methodology that allows the UI to be specified in UML diagrams formally using the OASIS specification language (Letelier et al., 1998); OVID (*Object, View and Interaction Design*) (Roberts et al., 1998), a methodology that uses a set of UI design techniques for objects developed by IBM; WISDOM (*Whitewater Interactive System Development with Object Models*) (Nunes, 2001), a methodological proposal for UML-based UI development; UMLi (Pinheiro, 2002), a UML notation extension for designing UIs. Neither of these works goes deeper in dynamic UIs (evolutive and adaptable). There are also works like (López-Jaquero et al., 2009a, 2009b; Sottet et al., 2007, 2008; Vanderdonckt, 2005; Cámara et al., 2007), but they do not use automatic composition of interfaces. In Breiner et al. (2009) a runtime adaptable UI method is proposed, but it does not use COTS and the adaptation is limited. However, all these works have become source of inspiration for the methodology.

Regarding COTS components, there are a number of software development methodologies, methods and techniques (e.g., Wallnau et al., 2002; Meyers & Obendorf, 2001; Heineman & Council, 2001). There are also publications where COTS are used for modular design of information system applications, such as the one in Alencar et al. (2002). In Iribarne et al. (2005) we developed a COTS composition experiment with commercial *Geographic Translator Service* (GTS) components used in *Geographic Information Systems* (GIS). In the last few years, the trend in the research of component-based software application development is to facilitate automatic integration of commercial components (COTS) through composition (or assembly) of their parts. To solve this task, we must determine a selection criterion that can be approached in different ways, intuitively, through direct assessment or indirect methods. In Leung and Leung (2003) the authors develop an indirect assessment method based on domains (domain-based COTS-product selection method: DBCS), which reduces the complexity and improves efficiency, by taking into consideration the relationships between the components and the system.

Although many studies use the COTS paradigm, few describe realistic information systems development cases making use of these components, and fewer still describe COTS multi-component user interfaces. Wagner and Nielsen (2008) present a product that makes use of COTS as clients of an activated Bluetooth device which uses these available components to construct a graphic interface. Such interface is originally empty and treats each component as an element of the interface that summarizes the functionality of the service that it offers. Another example of a UI constructed by as-

Figure 1. A piece of an iGoogle User Interface



sembling graphical components is the traditional iGoogle user interface (http://www.google.es/ig) (Figure 1); this interface offers an extensive catalogue of services that can be added to our personalized interface, bearing in mind that such elements are not dependent on each other and any combination is possible. For instance, the iGoogle UI contains five *widgets*: a calendar, a calculator, a translator, a weather service, and a Google map service.

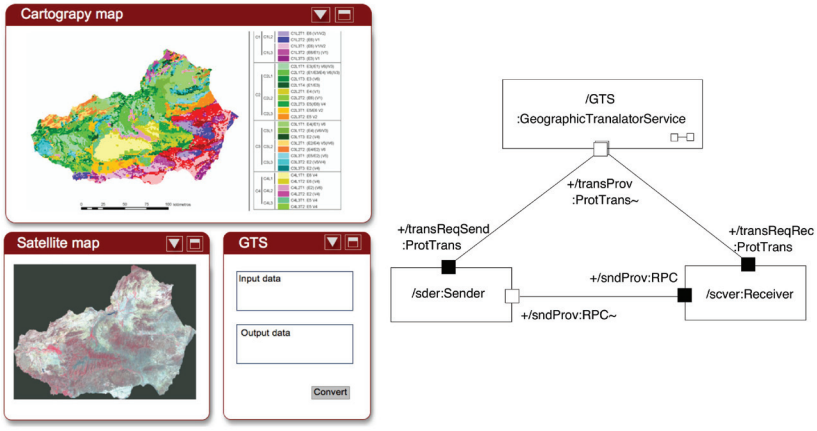
Our proposal is just inspired on the *widgets* components of an iGoogle interface. Due to their Web nature and simplicity, these interfaces (and components) are satisfactorily applicable in WIS environments as *Environmental Management Information Systems* (EMIS) (International Organization for Standardization, 1996).

3. A CASE STUDY

To explain the proposal let us suppose the scenario of a simple converter service of spatial images known as *Geographic Translator Service* (GTS). This type of service is very usual in distributed GIS and EMIS, where the whole of the information is distributed and interconnected using object-oriented techniques. Figure 2 shows the conceptual behaviour of a GTS written in UML-RT notation (right): a language for modelling complex real-time systems (e.g., telecommunications, automatic control and aerospace applications, among others). Although it is a visual modelling language with formal semantics for specifying, visualizing, documenting and automating the construction of complex, event-driven, and distributed real-time systems, we use this notation only as a conceptual way to describe the *cotsget architecture* because it uses simple notation. UML-RT notation uses: capsules, ports and connectors. Capsules represent a graphical notation to refer to the whole component. Ports mediate the interaction of the capsule with the outside world by means of small black box, to describe the component's provided behaviour, and small white boxes to describe those component's required behaviour. Connectors are represented by means of lines that unite two ports. If a connector unites more than one port, it must be duplicated in the capsule that requires it, shown as a double small box.

The behaviour of a GTS component is as follows. Using a message, the offering compo-

Figure 2. The Geographical Translator System (GTS) example



nent (Sender in the figure) delegates the conversion process to the GTS component. The message contains the conversion information. The translated image is stored in a buffer, and then the GTS returns an identifier to the origin component. After that, this id is sent to the requiring component, which uses it as a code to extract the converted file from the buffer.

Figure 2 also shows an example of user interface (left) composed by three cotsgets, those related to the GTS framework: a UI component that uses a cartographical map which must be converted into a satellite image map and shown in other UI component (in other area/portion of the user interface); In a third UI component the user sets the parameters for the conversion process, which can operate as an external service through the UI component.

4. COTSGET-BASED USER INTERFACES

Implementation of dynamic WIS UIs (such as the SOLERES) can be done by component composition. To do this, there must be a market (an industry) of UI components, known in the literature as a CBSD (*Component-Based Software Development*) as a COTS component. Based on this component market, and with the intervention of a trading process, the system will be able to generate the user interface based on established criteria: a user-interface architecture.

SOLERES-HCI includes the construction of simple WIMP user interfaces based on "bottom-up" composition of COTS interface components like *widgets* or *gadgets*. We call these user interface components *cotsgets*, and they could be resident in public repositories. Each UI component (*cotsget*) has a concrete functionality and is prepared for assembly and operation with others (Figure 3). Concrete *cotsget* components reside in trader repositories. Component's provided/required services are used by traders in selecting processes to fulfill the *dependences* requirements fixed by the *cotsget* architecture.

Definition 1 (concrete cotsget). *A concrete cotsget CG=(s,n,P,R) consists of:*

(i) a stereotype <<cotsget>>;
(ii) a component name n;
(iii) a finite set P of provided services $P = \{P_1, \ldots, P_n\}$;
(iii) and a finite set R of required services $R = \{R_1, \ldots, R_n\}$.

Each *cotsget* has a series of properties of dependency. Some of these properties are related to visual aspects, behavior, etc. The dependences

Figure 3. A WIS cotsget-based interface following a "bottom-up" perspective

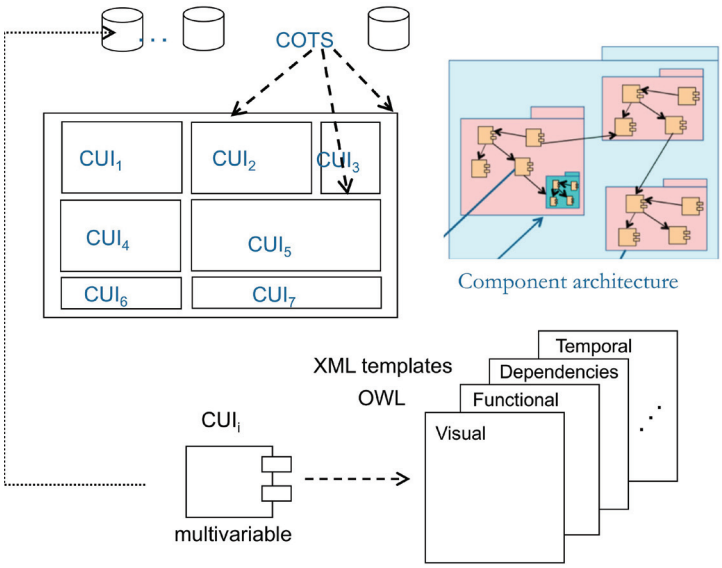
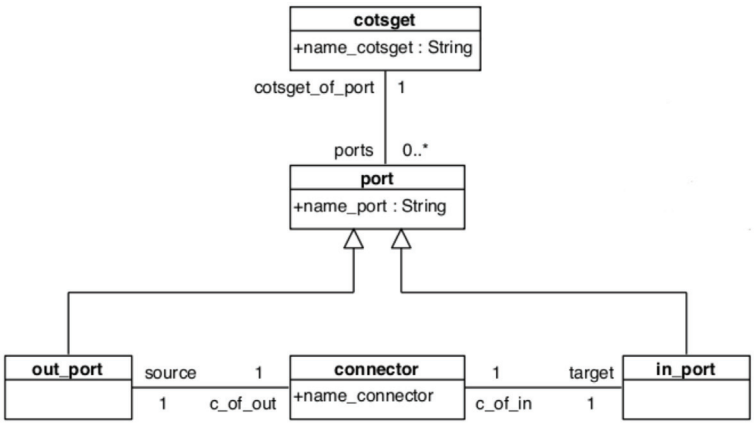


Figure 4. The meta-model of an architecture of cotsgets components



can be spatial, temporal, etc. These properties depend in turn on individual and/or group factors related to the system users. All of these aspects define the specification of each *cotsgets* and are described by a specification language. Therefore, the whole WIMP-*cotsgets* user interface will respect some composition principles (e.g., dependences between components, use restrictions, availability, visibility, etc.). In our case we will only consider dependences between component in a general way (i.e., for interoperability issues).

Definition 2. (Dependency). *A dependency D=(n,in,out) between cotsget component is a connector between two cotsget components which consists of:*

(i) a connector name n;

(ii) a connector rol "in" as a input to a provided service of cotsget;
(iii) and connector rol "out" as a output from a required service of cotsget.

Dependences between components are described in an architecture. We use UML notation to describe a profile for *cotsget* architectures. The methodology defines the *cotsget* architecture by means of metamodel, which is a conceptual representation written in UML class diagram.

Definition 3. (Cotsgets Architecture). *A cotsgets architecture A=(n,C,D) consists of:*
(i) an architecture name n;
(ii) a finite set C of cotsget components p,q,r…;
(iii) and a finite set D of dependences u,v,w,…, where $D \subseteq (C \times C)$.

As usual, we write $p \rightarrow q$, rather than $(p,q) \in \rightarrow$.

The profile was described through a UML meta-model shown in Figure 4. The meta-model establishes the rules and elements that describe an interface architecture through a model (an instance of the meta-model). Here, an interface architecture of *cotsgets* components defines different aspects not only of the interface itself (i.e., visual aspects, position, etc.) but also dependences between components, simple interaction (individual) or complex (cooperative), among other aspects. The present work is based only on dependences between components. Thus, we'll consider interface architecture as a set of components with dependences between them.

An interface component is a component which offers a service and may require others' service to work. This requirement is dependence in the architecture. For example, Figure 5 shows a user interface architecture for the scenario described in Section 3 and it visually corresponds to the interface in Figure 2. Dependences between the three components are unknown in the graphical user interface (Figure 2); however, in its associated model (Figure 5) the "Satellite" component is offering a service to the "Cartography" component. In addition, "GTS" is requiring some information from "Satellite" and "Cartography". Components are defined by means of an empty and stereotyped UML class like "<<cotsget>>".

Dependences are represented by a UML association (called "connector") having a name and two roles. Roles beginning with the word "out" represent a service offer (component's behaviour). The role name is created by adding the name of the component associated to the role and the name of the component that is connecting. For instance, a "satellite_gts" connector can link two cotsgets components called "satellite" and "gts". The order in the name indicates the dependence's direction. The role "in" means the component is requiring a service as an input. Roles "in" are represented as an arrow in the connector. The meta-model defines a *cotsget* component as a class with a name called "name_cotsget" and a set of ports that can be input/output. Figure 6 shows the textual XMI model of the example user interface.

Ports are roles of dependence relationship between components. There are two types of ports: input or output. The ports (roles) have a name "name_port". Dependence relationships are defined with a "connector" that always links an "out_port" with an "in_port". The meta-model declaration itself avoids dependences (connectors) like "out-out" or "in-in". The meta-model can be enriched with restrictions written in OCL of UML in order to avoid reflexive dependence relationships, that is, dependences between the same component. Such restriction upon the impossibility of association of a component with itself was included outside the metamodel, in order not to overload the example. Nevertheless, in MDD there is usually a "model-checking" process to validate whether models fulfill a meta-model. So, restrictions can either be directly included in the meta-model or imposed in the "model-checking" process.

To facilitate the samples, we developed a platform with different tools. One of them is an Eclipse GMF tool that allows drawing models

Figure 5. A model instance of the meta-model

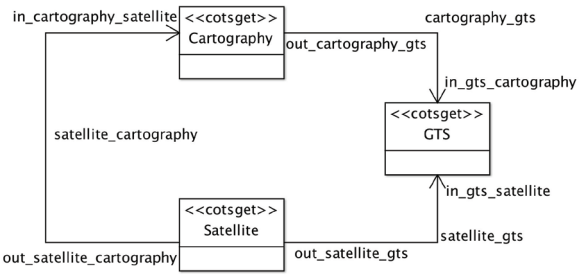


Figure 6. The XMI object-model of the user interface

```
<?xml version="1.0" encoding="ASCII"?>
<meta_model:model xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:meta_model="http://www.ual.es/acg/soleres/mt/meta_model.ecore">
  <connectors target="//@cotsgets.1/@ports.1" source="//@cotsgets.0/@ports.1" name_connector="satellite_cartography"/>
  <connectors target="//@cotsgets.2/@ports.0" source="//@cotsgets.0/@ports.0" name_connector="satellite_gts"/>
  <connectors target="//@cotsgets.2/@ports.1" source="//@cotsgets.1/@ports.0" name_connector="cartography_gts"/>
  <cotsgets name_cotsget="Satellite">
     <ports xsi:type="meta_model:out_port" name_port="out_satellite_gts" c_of_out="//@connectors.1"/>
     <ports xsi:type="meta_model:out_port" name_port="out_satellite_cartography" c_of_out="//@connectors.0"/>
  </cotsgets>
  <cotsgets name_cotsget="Cartography">
     <ports xsi:type="meta_model:out_port" name_port="out_cartography_gts" c_of_out="//@connectors.2"/>
     <ports xsi:type="meta_model:in_port" name_port="in_cartography_satellite" c_of_in="//@connectors.0"/>
  </cotsgets>
  <cotsgets name_cotsget="GTS">
     <ports xsi:type="meta_model:in_port" name_port="in_gts_satellite" c_of_in="//@connectors.1"/>
     <ports xsi:type="meta_model:in_port" name_port="in_gts_cartography" c_of_in="//@connectors.2"/>
  </cotsgets>
</meta_model:model>
```

of *cotsgets* architectures (as shown in Figure 5). This tool is mapping the graphical elements with the ones in object model (instantiated meta-model). The triplet (*out_port*, *connector*, *in_port*) of the meta-model (presented in the object model through three classes) is represented by an association (one line) in the model. The name attribute of the connector is mapped as association name and the name attributes of "out_port" and "in_port" classes are mapped as role names in the model.

Moreover, the arrow of the association line (dependence) is placed at the end of "in_port" class of the object model. For each dependence, a "connector" is created in the object diagram and, therefore, in the XMI template. See how each element of the object diagram is stored in its XMI template (Figure 6). The container element of an architecture XMI template is "meta_model:model", as an instance model of a meta-model. Out of three attributes of the container element, the first two make reference to names space for typeng, and the third one to the base meta-model "meta_model.ecore".

5. CONCLUSIONS AND FUTURE WORK

Nowadays there has been a special interest in the globalization of the information through a common vocabulary (i.e., ontologies), and the standardized way in which information is retrieved on the Web (i.e., powerful search engines, and intelligent software agents). The same principles of globalization and standardization should also be valid for the user interfaces of the *Web-based Information Systems* (WIS), but they are built on traditional development paradigms. In this paper we presented an approach inspired on a *Model-Driven Development* (MDD) perspective to reduce the gap of globalization/standardization in the composition of user interfaces from model and metamodel representations of *widgets*-type COTS interface components architectures.

As future work, we'll define an ontology of a user interface component in order to facilitate the generalization and specialization tasks of collections and types of interfaces. We'll make

http://doi.org/10.4018/ijksr.2011070106

the interfaces be a mutual element between groups of users in different places and working with the same interface for a decision-making in a WIS. In these cases, we may also suppose that what is shown in the user interface is different for some users, depending on the role or the user. For example, in our information system, we may have a user profile "politician" who is cooperating with another user profile "technician" for a common decision-making. A politician interface may have an interface portion (*gadget*) relating to money matters that the technician interface needn't show (and viceversa similarly). In our example, the users interface was the same.

Moreover, considering several user profiles cooperating with each other for decision-making through WIS and on the basis that these systems are often accessed by groups of users (with different profiles), we'll try to look into how to adapt such interfaces to each user's needs. Therefore, we'll investigate how the registered user interface (and therefore, known by the system) is being developed through time to be adapting itself to the needs of the user or group of user that cooperates with each other. We'll study the way of determining which information will allow the interfaces to develop through time.

ACKNOWLEDGMENT

This work has been supported by the EU (FEDER) and the Spanish MICINN under grant of the projects TIN2010-15588 and TRA2009-0309, and the project JUNTA ANDALUCIA (proyecto de excelencia) TIC-6114.

REFERENCES

Alencar, P. S. C., Cowan, D. D., & Luo, M. (2002). A framework for community information systems. *Annals of Software Engineering*, *13*(1), 381–411. doi:10.1023/A:1016518115185

Almendros, J., & Iribarne, L. (2008). An extension of UML for the modeling of WIMP user interfaces. *Journal of Visual Languages and Computing*, *19*(6), 695–720. doi:10.1016/j.jvlc.2007.12.004

Asadi, M., & Ramsin, R. (2008). MDA-based methodologies: An analytical survey. In I. Schieferdecker & A. Hartman (Eds.), *Proceedings of the 4th European Conference on Model-Driven Architecture–Foundations and Applications* (LNCS 5095, pp. 419-431).

Asensio, J. A., Iribarne, L., Padilla, N., Muñoz, F. J., Ayala, R. M., Cruz, M., & Menenti, M. (in press). *A MDE-based satellite ontology for environmental management systems*. New York, NY: Springer.

Bisconti, C., Corallo, A., De Maggio, M., Grippa, F., & Totaro, S. (2010). Quantum modeling of social dynamics. *International Journal of Knowledge Society Research*, *1*(1), 1–12. doi:10.4018/jksr.2010010101

Breiner, K., Görlich, D., Maschino, O., Meixner, G., & Zühlke, D. (2009). Run-time adaptation of a universal user interface for ambient intelligent production environments. In J. A. Jacko (Ed.), *Proceedings of the 13th International Conference on Human-Computer Interaction* (LNCS 5613, pp. 663-672).

Cámara, J., Canal, C., Cubo, J., & Murillo, J. M. (2007). Enabling adaptivity in user interfaces. In F. Oquendo (Ed.), *Proceedings of the First European Conference on Software Architecture* (LNCS 4758, pp. 106-114).

Cranefield, S., & Purvis, M. (2001). Integrating environmental information: Incorporating metadata in a distributed information systems architecture. *Advances in Environmental Research*, *5*, 319–325. doi:10.1016/S1093-0191(01)00082-X

Dance, S., Gorman, M., Padgham, L., & Winikoff, M. (2003). An evolving multi agent system for meteorological alerts. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems* (pp. 966-967).

Heineman, T. H., & Council, W. T. (2001). *Component-based software engineering: Putting the pieces together* (pp. xlii, 818). Reading, MA: Addison-Wesley.

Hernández-López, A., Colomo-Palacios, R., García-Crespo, A., & Soto-Acosta, P. (2010). Trust building process for global software development teams: A review from the literature. *International Journal of Knowledge Society Research*, *1*(1), 1–12. doi:10.4018/jksr.2010010105

International Organization for Standardization. (1996). *ISO 14001: Environmental management systems-Specification with guidance for use*. Retrieved from http://www.iso.org/iso/iso_14000_essentials

http://doi.org/10.4018/ijksr.2011070106

International Organization for Standardization. (1997). *ISO/IEC13235: ODP trading function specification.* Retrieved from http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=21470

Iribarne, L., Padilla, N., Asensio, J. A., Muñoz, F. J., & Criado, J. (2010). Involving web-trading agents & MAS: An implementation for searching and recovering environmental information. In *Proceedings of the International Conference of Agents and Artificial Intelligence*, Valencia, Spain (pp. 268-273).

Iribarne, L., Troya, J. M., & Vallecillo, A. (2004). A trading service for COTS components. *The Computer Journal*, *4*(3), 342–357. doi:10.1093/comjnl/47.3.342

Iribarne, L., Troya, J. M., & Vallecillo, A. (2005). Trading for COTS components to fulfill architectural requirements . In De Cesare, S., Lycett, M., & Macredie, R. D. (Eds.), *The development of component-based information systems* (pp. 202–222). Armonk, NY: M.E. Sharpe.

Letelier, P., Ramos, I., Sánchez, P., & Pastor, O. (1998). *OASIS version 3: A formal approach for object oriented conceptual modeling*. Madrid, Spain: OASIS.

Leung, H. K. N., & Leung, K. R. P. H. (2003). Domain-based COTS-product selection method. In A. Cechich, M. Piattini, & A. Vallecillo (Eds.), *Component-Based Software Quality: Methods and Technology* (LNCS 2693, pp. 40-63).

López-Jaquero, V., Montero, F., & Gonzalez, P. (2009). AB-HCI: An interface multi-agent system to support human-centered computing. *IET Software*, *3*(1), 14–25. doi:10.1049/iet-sen:20070108

López-Jaquero, V., Montero, F., & Real, F. (2009). Designing user interface adaptation rules with T:XML. In *Proceedings of the 13th International Conference on Intelligent User Interfaces* (pp. 383-388).

Lozano, M. D., Ramos, I., & González, P. (2000). User interface specification and modeling in a object oriented environment for automatic software development. In *Proceedings of the International Conference on Technology of Object-Oriented Languages and Systems* (pp. 373-381).

Meyers, B. C., & Obendorf, P. (2001). *Managing software acquisition: Open systems and COTS products* (pp. xxvii, 360). Reading, MA: Addison-Wesley.

Nunes, N. (2001). *Object modeling for user-centered development and user interface design: The WISDOM approach.* Unpublished doctoral dissertation, University of Madeira, Madeira, Portugal.

OMG. (2008). *MOF 2.0: Query/views/transformations RFP.* Retrieved from http://www.omg.org/techprocess/meetings/schedule/MOF_2.0_Query_View_Transf._RFP.html

Padilla, N., Iribarne, L., Asensio, J. A., Muñoz, F. J., & Ayala, R. (2008). Modelling an environmental knowledge-representation system. In M. D. Lytras, J. M. Carroll, E. Damiani, & R. D. Tennyson (Eds.), *Proceedings of the First World Summit on Emerging Technologies and Information Systems for the Knowledge Society* (LNCS 5288, pp. 70-78).

Pinheiro, P. (2002). *Object modelling of interactive systems: The UMLi approach.* Unpublished doctoral dissertation, University of Manchester, Manchester, UK.

Roberts, D., Berry, D., Isensee, S., & Mullaly, J. (1998). *Designing for the user with OVID: Bridging user interface design and software engineering*. Indianapolis, IN: New Riders Publishing.

Sharma, R. S., Ng, E. W. J., Dharmawirya, M., & Samuel, E. M. (2010). A policy framework for developing knowledge societies. *International Journal of Knowledge Society Research*, *1*(1), 1–12. doi:10.4018/jksr.2010010103

Sottet, J. S., Calvary, G., Coutaz, J., & Favre, J. M. (2008). A model-driven engineering approach for the usability of plastic user interfaces. In J. Gulliksen, M. B. Harning, P. Palanque, G. C. van der Veer, & J. Wesson (Eds.), *Proceedings of the Joint Working Conferences on Engineering Interactive Systems* (LNCS 4940, pp. 140-157).

Sottet, J. S., Ganneau, V., Calvary, G., Coutaz, J., Demeure, A., Favre, J. M., & Demumieux, R. (2007). Model-driven adaptation for plastic user interfaces. In C. Baranauskas, P. Palanque, J. Abascal, & S. DinizJunqueira Barbosa (Eds.), *Proceedings of the 11th IFIP TC 13 International Conference on Human-Computer Interaction* (LNCS 4662, pp. 397-410).

Stahl, T., & Völter, M. (2006). *Model-driven software development*. New York, NY: John Wiley & Sons.

UML. (2005). *Unified modeling language specification, version 2.0.* Needham, MA: Object Management Group (OMG).

Vanderdonckt, J. (2005). A MDA-compliant environment for developing user interfaces of information systems. In O. Pastor & J. F. e Cunha (Eds.), *Proceedings of the 17th International Conference on Advanced Information Systems Engineering* (LNCS 3520, pp. 16-31).

Wagner, S., & Nielsen, C. C. (2008). Usability and implementation experiences with COTS products used as a distributed client platform. In *Proceeding of the 3rd International Conference on Pervasive Computing and Applications* (pp. 399-404).

Wallnau, K. C., Hissam, S. A., & Seacord, R. C. (2002). *Building systems from commercial components (*pp. xv, 379). Reading, MA: Addison-Wesley.

Luis Iribarne received the B.S. and M.S. degrees in computer science from the University of Granada, and the Ph.D. degree in computer science from the University of Almeria, Spain. From 1991 to 1993, he worked as a Lecturer at the University of Granada, and collaborated as IT Service Analyst at the University School of Almeria. Since 1993, he has been a Lecturer in the Advanced College of Engineering at the University of Almeria. From 1993 to 1999, he worked in several national and international research projects on distributed simulation and geographic information systems. He has been the coordinator of several projects founded by the Spanish Ministry of Science and Technology (MICINN). He is also the Research Chief of the Applied Computing Group. His research interests include software engineering, model-driven engineering, ontology-driven engineering, component-based software development, COTS components, trading, agents and multi-agent systems, and UML design. Dr. Iribarne is a member of IEEE and EUROMICRO.

Nicolas Padilla received the B.S. and Ph.D. degrees in computer science from the University of Almeria in Spain, and the M.S. degree in computer science from the University of Granada, Spain. From 1991 to 1993, he worked as an Associate Professor at the University of Granada. Since 1993, he has worked as Associate Professor in the Advanced College of Engineering at the University of Almeria. In 2007, he co-founded the Applied Computing Group. His research interests include cooperative systems, model-driven engineering, human-computer interaction, agents and multi-agent systems, and UML design.

José Andrés Asensio received the B.S. and M.S. degrees in computer science from the University of Almeria, Spain, and he is currently a doctoral student at the University of Almeria (UAL). Since 2006, he worked in three research projects on Environmental Management Systems. In 2007 he joined Applied Computing Group and in 2009, he became Associate Professor at the University of Almeria. Since 2003 he works as Technician in the Unit of Support Technologies for Learning and Virtual Learning at the University of Almeria. His research interests include trading, agents and multi-agent systems, ontology-driven engineering, model-driven engineering, model transformations and system modelling.

http://doi.org/10.4018/ijksr.2011070106

Javier Criado received the B.S. and M.S. degrees in computer science from the University of Almeria, Spain, where he is also currently a doctoral student. Since 2009, he has worked as a Researcher in the project SOLERES of the Spanish Ministry of Science and Innovation. In 2009, he joined the Applied Computing Group. His research interests include model-driven engineering, model transformations, ontology-driven engineering, human-computer interaction for advances on user interfaces, model-driven development for advanced user interfaces, COTS components, trading, agents and multi-agent systems, and UML

http://doi.org/10.4018/ijksr.2011070106