Proceedings of the Fifth International Workshop on Web-Oriented Software Technologies

IWWOST.05

Porto, Portugal. June 13, 2005

PREFACE

Welcome to IWWOST 2005, the fifth International Workshop on Web-Oriented Software Technologies.

Since 2001, IWWOST has been an international forum for discussing state-of-the-art modelling approaches, methods and technologies for Web applications. The first edition was held in Valencia in 2001, the second in Málaga (together with ECOOP 2002), the third in Oviedo and the fourth in Munich (both co-located with ICWE 2003 and ICWE 2004 respectively).

IWWOST has been conceived as a place for methodologists, designers and developers to meet and exchange their experiences in the process of building complex Web applications. Usually, IWWOST attendees work on the same problem from different points of view, those supported by the method or design approach of their choice. In this way, IWWOST participants can compare their own approaches with others' and discuss strengths and weaknesses of each approach. Most widely known methods (like WebML, WSDM, OOHDM, OO-H, UWE, WUML, W2000, etc) have been discussed in previous IWWOST editions.

This year we have an exciting program which is organized in three sessions: First session is dedicated to Adaptive Web and Usability, second session covers issues such as Web Services and Integration and finally, last session is related to Semantic Web technologies. Moreover, as the IWWOST workshop is oriented towards discussion, we have planed to have two panels, the former at the end of session 1 and the latter to close the event at the end of afternoon sessions. We hope you will enjoy the workshop.

June 2005

Daniel Schwabe
Gustavo Rossi
Luis Olsina
Vicente Pelechano

WORKSHOP ORGANIZERS

Daniel Schwabe, BRAZIL
Pontifícia Universidade Católica do Rio de Janeiro

Gustavo Rossi, ARGENTINA
Universidad Nacional de la Plata

Luis Olsina, ARGENTINA
Engineering School at UNLPam

Vicente Pelechano, SPAIN
Technical University of Valencia

LOCAL EDITORS

Victoria Torres, SPAIN
Technical University of Valencia

Gonzalo Rojas, SPAIN
Technical University of Valencia

PROGRAM

09.00 - 09.15 Registration

09.15 - 09.30 Opening Session

09.30 - 10.30 Session 1 Adaptative Web & Usability

Authors	Article
Gonzalo Rojas, Vicente Pelechano and Joan Fons	A Model-Driven Approach to include Adaptive Navigational Techniques in Web Applications
Vito Perrone, Luca Mainetti and Paolo Paolini	A UML Extension for Designing Usable User Experiences for Web Applications

10.30 - 10.45 Coffee Break

10.45 - 12.15 Session 1 Adaptative Web & Usability

Authors	Article
Marco Winckler and Jean Vanderdonckt	Towards a User-Centered Design of Web Applications based on a Task Model
Andreas Papasalouros and Symeon Retalis	Defining a UML Profile for Web–based Educational Applications
Luisa Mich, Mariangela Franch	Instantiating Web Sites Quality Models: an Ontologies driven Approach

12.15 - 12.45 Discussion Panel

12.45 - 14.30 Lunch

14.30 - 16.00 Session 2 Web Services & Integration

Authors	Article
Daniel Sanz, Ignacio Aedo and Paloma Díaz	A Web Service for Hypermedia Role-Based Policies
Daniela Leal Musa, Marcel Weschenfelder, José Palazzo Moreira de Oliveira	Developing an Alarm Manager Based on Web Services
Nathalie Moreno and Antonio Vallecillo	Modeling Interactions between Web Applications and Third Party Systems

16.00 – 16.30 Coffee Break

16.30 - 17.30 Session 3 Semantic Web

Authors	Article
Laura Montells, Susana Montero, Paloma Díaz and Ignacio Aedo	Model-Driven Approach for Semantic Web based-hypermedia Applications
Peter Barna, Geert-Jan Houben, Philippe Thiran, Ad Aerts and Flavius Frasincar	Reusable Navigation Templates to Support Navigation Design in Hera

17.30 - 18.00 Closing Discussion

PARTICIPANTS

Ad Aerts, Technische Universiteit Eindhoven, The Netherlands
Andreas Papasalouros, National Technical University of Athens, Greece
Antonio Vallecillo, Universidad de Málaga, Spain
Symeon Retalis, University of Piraeus, Greece
Daniela Leal Musa, Instituto de Informática - UFRGS
Daniel Sanz, Universidad Carlos III de Madrid, Spain
Flavius Frasincar, Technische Universiteit Eindhoven, The Netherlands
Geert-Jan Houben, Technische Universiteit Eindhoven, The Netherlands
Gonzalo Rojas, Technical University of Valencia, Spain
Ignacio Aedo, Universidad Carlos III de Madrid, Spain
Jean Vanderdonckt, Université catholique de Louvain, Belgium
Joan Fons, Technical University of Valencia, Spain
José Palazzo Moreira de Oliveira, Instituto de Informática – UFRGS, Brazil
Laura Montells, Universidad Carlos III de Madrid, Spain
Luca Mainetti, Università degli Studi di Lecce, Italy
Luisa Mich, University of Trento, Italy
Marcel Weschenfelder, Instituto de Informática - UFRGS, Brazil
Marco Winckler, LIIHS-IRIT Toulouse,France
Mariangela Franch, University of Trento, Italy
Nathalie Moreno, Universidad de Málaga, Spain
Vicente Pelechano, Technical University of Valencia, Spain
Paloma Dias, Universidad Carlos III de Madrid, Spain
Peter Barna, Technische Universiteit Eindhoven, The Netherlands
Philippe Thiran, Technische Universiteit Eindhoven, The Netherlands
Susana Montero, Universidad Carlos III de Madrid, Spain
Vito Perrone, Paolo Paolini, Politecnico di Milano, Italy

TABLE OF CONTENTS

A Model-Driven Approach to Include Adaptive Navigational Techniques in Web Applications

Gonzalo Rojas, Vicente Pelechano and Joan Fons
Department of Information Systems and Computation
Technical University of Valencia, Spain
{grojas, pele, jjfons@dsic.upv.es}

Abstract

Adaptivity is an increasingly demanded characteristic of Web applications. However, adaptive techniques usually implemented in Adaptive Hypermedia Systems have been hardly considered by current Model-Driven Web Development Methods. This work presents an approach to describe adaptive navigation techniques at early stages of the Web development process. Using the primitives of the OOWS Navigational Model, we have defined a strategy to incorporate three types of techniques (link-hiding, link-ordering and link-annotation) to the Web modelling process. A User Modelling proposal that is necessary for the complete specification of these techniques is also introduced. The impact of the introduced descriptions of techniques on the final adaptive applications is shown by means of a case study.

1. Introduction.

Most of the currently available hypermedia applications provide users with a limited set of possible interactions, with little consideration of the particular characteristics, preferences and needs of each user.

Adaptive Hypermedia arises as an approach that seeks to achieve usage experiences that are more suitable to the individual information needs. Research efforts made by Adaptive Hypermedia community have produced clearly defined concepts and adaptation strategies. However, they are mainly focused on late stages of software development. The developed applications are highly dependant on implementation details and on particular data instances.

As a special and widespread kind of hypermedia systems, Web applications are incorporating adaptivity features. The Model-Driven Web development community has been augmenting the expressiveness of their Web conceptual models for achieving high-level descriptions of Adaptive Web Systems.

However, most of these efforts are limited to the definition of rules that constrain the accessibility of the navigational structures. This strategy gives little support to the implementation of well known adaptive techniques that have been already implemented in existing Adaptive Hypermedia Systems (AHS). The classification of techniques for adaptive navigation and presentation that has been proposed by the Adaptive Hypermedia community is hardly considered and it is not expressed at conceptual level, taking little advantage of this previous research efforts.

The main contribution of this work is introducing descriptions of well-known adaptive techniques at conceptual level of the Web development process. In this way, we are able to obtain adaptive Web applications that are more independent of implementation concerns than the most adaptive hypermedia systems.

In the context of OOWS [5], an OO-based modelling method, the specific contributions of this work are:
(a) A User Modelling approach that allows describing the intended users through the definition of groups of similar users (stereotypes) and three graphical models, which contains the needed attributes and operations for achieving the adaptivity.
(b) A set of adaptive navigation techniques that are defined in terms of different conceptual primitives of the OOWS Navigational Model.

The adaptive techniques that are introduced in the OOWS Web development method are classified into three categories that are broadly used by the Adaptive Hypermedia community: (a) the management of nodes accessibility (*link-hiding*); (b) the ordering of the navigational links (*link-ordering*); and (c) the addition of informative hints to displayed links (*link-annotation*).

The rest of this paper describes the three parts of our proposal: Section 2 presents an overview of the OOWS Modelling approach, focused on its Navigational Model. Section 3 describes our proposal for User Modelling, consisting in the definition of user stereotypes and the specification of user attributes through three graphical models. Section 4 describes the adaptive navigation techniques introduced into the OOWS process, presenting examples of their conceptual description and the resulting implementations. Section 5 presents a brief review of related works on conceptual modelling of adaptive Web applications. Finally, Section 6 presents some conclusions and future works.

2. The OOWS navigational modelling approach.

OOWS (Object-Oriented Web Solution) modelling method [5] is the extension of the OO-Method proposal for Automatic Code Generation [7], which introduces the required expressiveness to capture the navigational and presentational requirements of web applications.

The OOWS Conceptual Modelling process consists of three main steps:
(1) *Requirements Elicitation*, where the requirements of the Web application are described by means of UML Use Cases and Scenarios techniques;
(2) *Classic Conceptual Modelling*, where system structure and behaviour are described, using UML-compliant Class, Sequence and State Diagrams; and
(3) *Navigational and Presentational Modelling*, where OOWS Navigational and Presentational Diagrams are built, strongly based on the Class Diagram defined in the previous stage.

The *OOWS Navigational Model* is composed of a set of *Navigational Maps*. Each map corresponds to a global view of the Web application for a given group of users. It is represented by a directed graph, in which the nodes are *Navigational Contexts* and the arcs are *Navigational Links* that define the valid navigation paths.

Navigational Contexts represent interaction points between users and application, and provide a set of cohesive data and operations. There are two types of navigational contexts: (a) *Exploration Contexts*, which are reachable from any node; and (b) *Sequence Contexts*, accessible only via predefined navigational paths.

Figure 1 shows an example of a Navigational Map, corresponding to an online bookstore and defined to a *Client* user type. It is composed of five exploration contexts (with "E" label) and three sequence contexts (with "S" label). Dashed arrows represent the navigational links to the exploration contexts. The solid arrows correspond to the predefined navigational paths that allow accessing the Books Navigational Context, and from this accessing to the other sequence contexts.

Each Navigational Context is composed of *Navigational Classes*, which represent views of the classes included in the Class Diagram. Each navigational context has one mandatory navigational class from which the information retrieval starts, called *Manager Class*, and other optional navigational classes that provide complementary information, called *Complementary Classes*.

Navigational classes contain the visible attributes and executable operations that are available for the user in the corresponding context. All the navigational classes are related through unidirectional binary relationships, so called *Navigational Relationships*. Each of them is defined over an association, aggregation, composition or specialization relationship included in the Class Diagram.

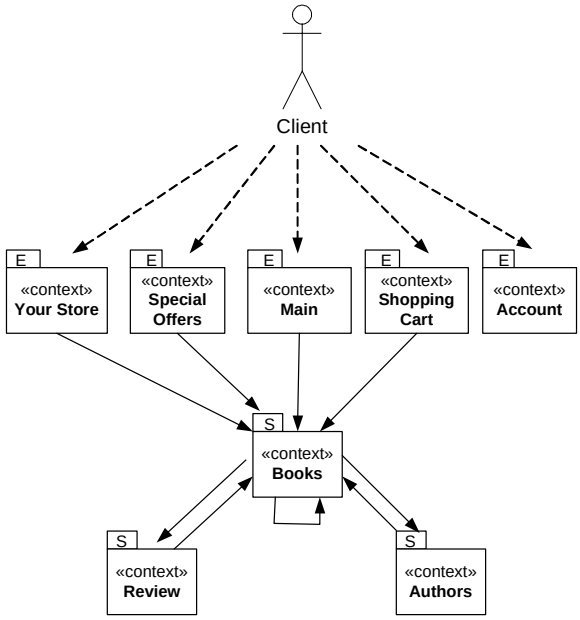


Figure 1. Example of OOWS navigational map and the two types of navigational contexts.

OOWS defines two kinds of Navigational Relationships:
(a) *Context dependency relationship* (dashed arrows), which represents a basic information recovery by crossing a structural relationship between classes.
(b) *Context relationship* (solid arrows), which represents an information recovery plus a navigation to a target context. Context relationships have the following properties:

(1) A *context attribute* that indicates the target context of the navigation (depicted as [*target context*]).

(2) A *link attribute* that specifies the attribute (usually one of the target navigational class) used as the "anchor" to activate the navigation to the target context.

Operation links can also be attached to an operation. An operation link represents the target context (depicted as [*target context*]) that the user will reach after that operation's execution.

Figure 2 shows the *Books* Navigational Context included in Figure 1, which describes a page of a specific book in the online bookstore. *Book* manager class shows basic purchase data of the presented book, while information from complementary classes is retrieved through the shown navigational relationships.

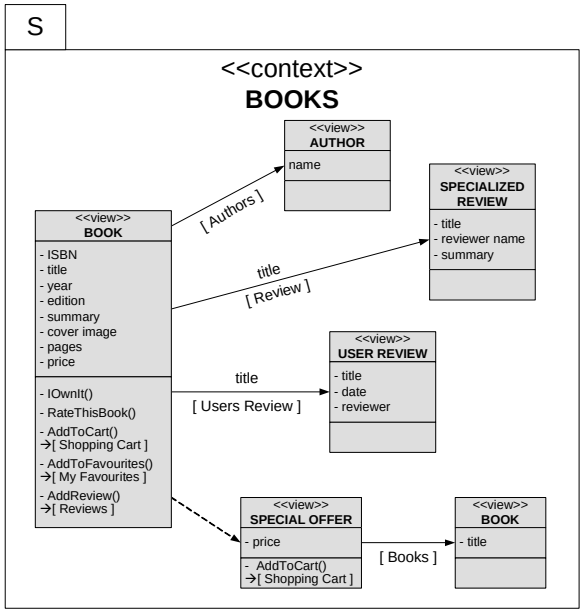


Figure 2. Books navigational context for a web-based bookstore.

Figure 3 shows an implementation of the *Books* Navigational Context. In frame 1, it is possible to distinguish the main data of the presented book, corresponding to attributes and operations of the *Book* manager class; the book review made by a specialist is shown through the *Book-Specialized Review* (see frame 2); book reviews made by other readers are shown through the *Book-User Review* (see frame 3); the data retrieved through the *Book-Special Offer* relationship are shown in frame 4 and correspond to a special offer (so-called "two for one" offer) related to the presented book.

The labels at the top of the page are anchors of links to other Exploration Contexts. Those links and their target contexts are always available. Meanwhile, the contexts that are accessible through the link of the book's author (frame 1), the link of the full-version of the specialized review (2) and through the links to the other readers' reviews (3) are contained in the Sequence Contexts (*Author* and *Review*, see Figure 1) and they are only accessible following a predefined path.

It is possible to define *filters* associated to a navigational class, which allows constraining the objects of this class that are retrieved through a navigational relationship.

As a presentational feature, OOWS includes the *Ordering Pattern*, which allows defining a certain order by which data of the navigational class are presented to the user. This order is based on some attribute of the manager class. The ordering can be ascendant or descendant.

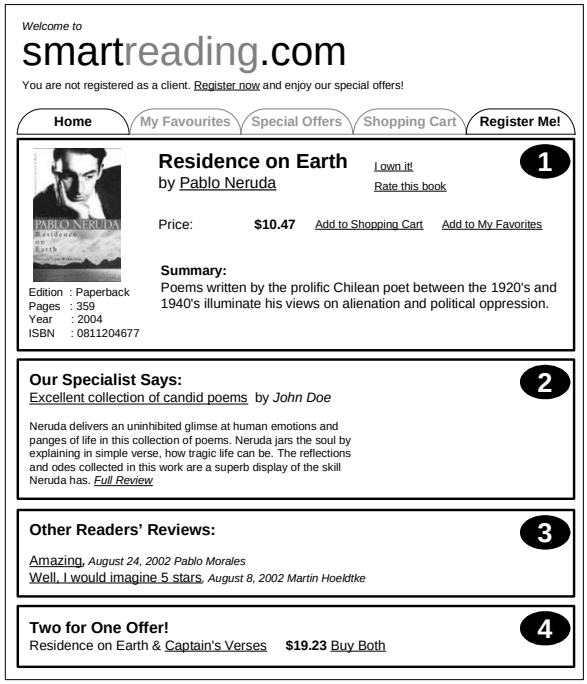


Figure 3. Implementation of Books navigational context.

Let us consider the following example: the section "Your Store" of an online bookstore shows some books that the system recommends to the user. These books must have been published not earlier than 1980 and must be presented in alphabetical order.

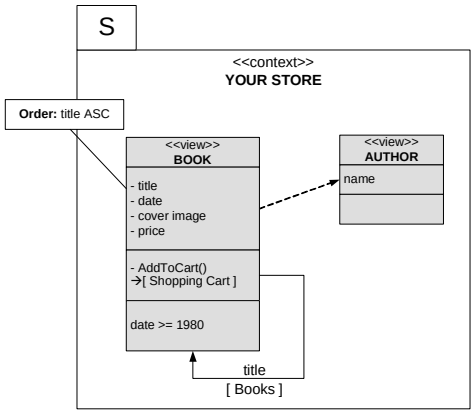


Figure 4. Navigational context for "Your Store" functionality.

The navigation description of this requirement is done through the context shown in Figure 4. It includes a filter (lower section of the *Book* manager class), which allows constraining the books to be displayed to those whose *date* attribute value is greater or equal than 1980.

Furthermore, the Ordering Pattern is applied to the manager class (left-side label), which indicates that the instances of this class must be ordered in ascendant order by their corresponding values of the *title* attribute. Figure 5 shows an implementation of these features.

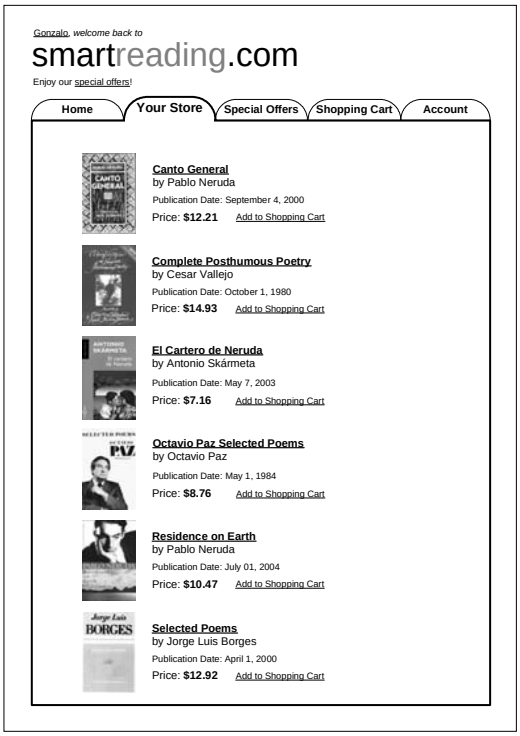


Figure 5. Implementation of "Your Store" functionality.

The conceptual tools of the OOWS Navigational Model provide multiple alternatives for the navigational design of the same web application depending on the user. Distinct navigational maps and variants of the inner structure of the intended contexts (nodes) can be associated to different kinds of users. For this reason, this model is very suitable to incorporate adaptive features to its descriptions. To achieve this goal, first we need a clear description of the intended application's users. In the next section, we present our proposal of User Modelling.

3. Modelling the user.

The goal of our User Modelling proposal is to obtain high level descriptions of the users of the Web application for achieving the system adaptation. We introduce two main steps:

(a) *Definition of User Stereotypes*, where the developer defines a set of user groups (stereotypes) that are ordered in a hierarchical structure;

(b) *Definition of User Model diagrams*, where three user diagrams are built, describing users in terms of: personal information; their relations with a particular application domain; and the navigational actions performed at execution-time.

In these steps, we apply two concepts broadly used for user's descriptions in existing Adaptive Hypermedia Systems. These concepts are:

(a) *Stereotypes*: This concept allow distinguishing several typical or "stereotypes" users, defining user groups whose members are likely to have certain homogeneous application-relevant characteristics [1]. Typical stereotypes are: skilled / novice users, child / young /adult clients.

(b) *Overlay Model*: It represents an individual user's knowledge of the subject as an "*overlay*" of the domain model. For each domain model concept, an individual overlay model stores some value, which is an estimation of the user knowledge level of this concept [8].

The *Stereotype* concept is the basis of the first step, while the *Overlay Model* is applied to the second one. Our proposal uses Overlay Models to describe not only the user's knowledge about a concept, but also other domain-dependent relationships between users and concepts, such as preferences and assigned rankings to a given concept.

3.1. Step 1: Definition of user stereotypes.

In this step, a set of user groups or user stereotypes are defined, according to the adaptation needs of the application. This classification allows taking benefits of some features that some users could share. Thus, the same implementation of an adaptive technique can be used by more than one specific user.

In the process of stereotypes specification, the developer must fulfil three tasks (adapted from [1]):

(a) *Stereotypes identification*: According to the application requirements, the developer identifies the subgroups of users that are likely to share some features and which can interact with the system. These subgroups are called *stereotypes*.

(b) *Stereotypes ordering:* Once defined, stereotypes are hierarchically ordered. These hierarchies allow avoiding redundancy in the definition of similar stereotypes, by inheritance of common attributes.

(c) *Identification of key user features*: The developer determines some key user features that describe the defined stereotypes, along with the values of these features that allow deciding the inclusion of users into each stereotype.

For instance, let us suppose that the online bookstore must distinguish the followings stereotypes: USER (anyone accessing the bookstore); CLIENT (registered user); CHILD, TEEN, YOUNG ADULT and ADULT

(according to his age); INFREQUENT READER, OCCASSIONAL READER and FREQUENT READER (according to his reading habits).

These stereotypes are hierarchically organized. Afterwards, for including users into these stereotypes, the following user characteristics are defined: for USER, no special feature is needed (it is the most general stereotype); CLIENT asks users to execute the *register()* operation. For determining whether a user belongs to CHILD, TEEN, YOUNG ADULT or ADULT stereotypes, a *birthdate* attribute is defined; finally, a *readingHabit* attribute describes how frequently a given user reads. The resulting stereotype hierarchy, along with the values of the attributes that describe each stereotype are shown in Figure 6.

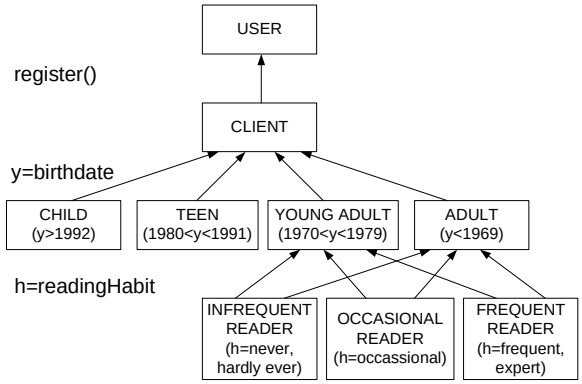


Figure 6. Example of user stereotypes hierarchy.

The definition of Stereotypes is a useful tool for classifying the application users for adaptive purposes. However, their expressiveness is limited, so the adaptive techniques must consider not only the attributes that describe the stereotype, but also the rest of the data expressed through the three diagrams of our proposed User Model.

3.2. Step 2: Definition of User Diagrams.

In this step, users are described through three UML-compliant Class Diagrams, which represent the three most used categories of User Modelling in AHS (examples can be found in [8] and [9]). The diagrams are the following:
(a) *Domain-Independent User Diagram*, which captures the user's personal data;
(b) *Domain-Dependent User Diagram*, which contains the user data related to the application domain; and
(c) *Navigational Behaviour Diagram*, which allows collecting data about the navigational actions that the user performs during a session.

The modeller classifies the attributes and operations that have been previously introduced in the stereotypes definition step, distinguishing those that depend of the application domain from the independent ones. These features are the first structures of the corresponding User Model Diagrams, which are completed with the conceptual structures needed for the fulfilment of the adaptivity requirements.

3.2.1. ***Domain-Independent User Diagram***. This diagram describes those personal user characteristics that have some relevance for the adaptation goals of the application. The values of the modelled data are independent of the user interaction with a particular Web application or specific application domains. For instance, the set of considered attributes may include *name*, *date of birth* or *city of residence*. Figure 7 shows an example of this diagram.

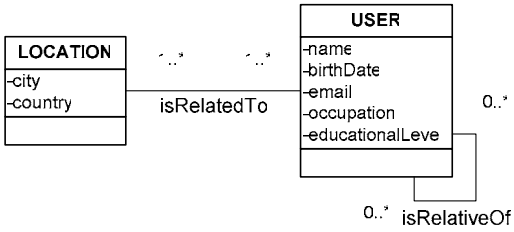


Figure 7. Example of a Domain-Dependent User Diagram.

3.2.2. ***Domain-Dependent User Diagram***. The user attributes and operations that are related to the application domain, along with their relationships with specific domain concepts, are specified in this diagram as an *overlay model* of the previously built structural diagram (Class Diagram).

From the concepts included in this diagram, the system can establish how relevant the domain concepts are for a given user. To achieve this, a *Relevance* property must be defined. This definition depends on the specific application domain and the adaptivity requirements. For instance, in an e-learning scenario, the relevance can be defined as the suitability of contents according to a learning goal, whereas in an e-commerce application the relevance may correspond to the similarity degree of a product to others previously purchased, considering the evaluation of these products that are explicitly assigned by the own user.

To model the relevance, the modeller should take into account specific functionalities of the application, such as the ranking assigned to some article on sale by the user, the purchase of the article or the learning unit the student has visited. The *Relevance* property is used to define the adaptive rules for navigational context instances. A more detailed description of the relevance property is presented in Section 4.

Figure 8 shows the Domain-Dependent User Diagram of the mentioned online bookstore. In the example, the designer pays special attention to the *Client-Book* relationships: books that the client has *visited* (i.e., has accessed their web pages); book titles that the user already *owns*; those ones he mostly prefers (*my_favourites*); and those ones in which he is *not interested*. Numerical values are used for describing the *ranking* assigned to some titles by the user.

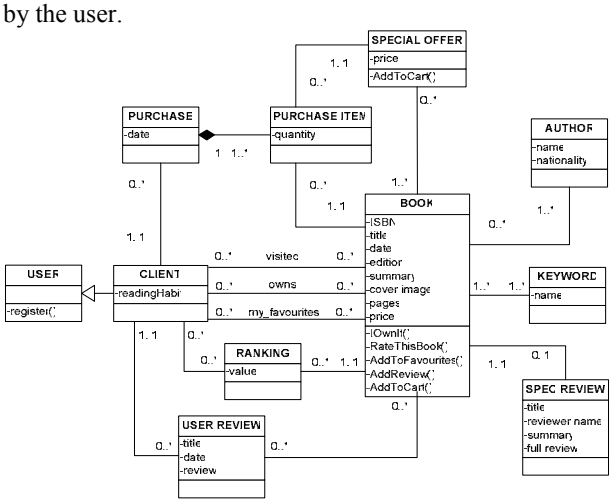


Figure 8. Domain Dependent User Diagram for an online bookstore.

3.2.3. ***Navigational Behaviour Diagram***. It describes the user's interaction with navigational structures of the application during a session. It is defined as an *overlay model* over the navigational structures that are specified for the Web application. This allows describing a user in terms of his visits to a given navigational context, the activations of links (context relationships) and the operations that he executes.

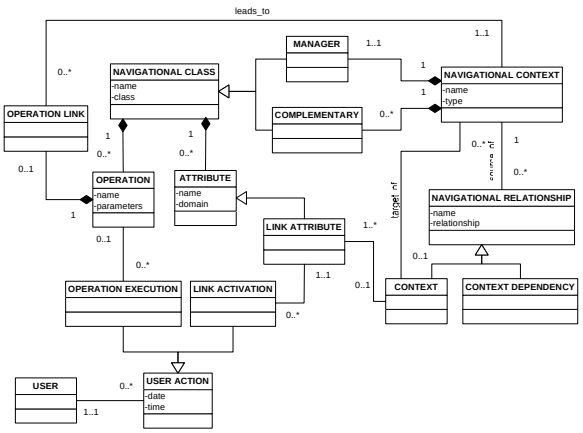


Figure 9. Example of OOWS Navigational Behaviour Diagram.

Figure 9 shows an example of this diagram. It is composed of a view of the OOWS navigational meta-model (which allow modelling the navigational structures), the *User* class and the *User Action* class, along its corresponding subclasses, which describe the activation of a link or the execution of an operation.

These data are used to update the User Model, specifically the Domain-Dependent User Diagram. Specific visited contexts and activated links can reveal important features of the user, such as the knowledge level about the accessed contents or the preferences for them. This update process can be achieved through update rules, whose definition is one of the future works of our proposal.

4. Adaptive techniques in the OOWS navigational model.

OOWS Navigational Model provides the developer with the required expressiveness to define adaptive navigation techniques at conceptual level. There are multiple techniques of adaptive navigation which are implemented in existing Adaptive Hypermedia Systems. Brusilovsky [9] classifies them into the following six groups:

a) *Adaptive Link Hiding*: these techniques restrict the navigation space by hiding the links to pages that are irrelevant or forbidden to a certain kind of users. The "hiding" can be implemented, for instance, by deactivating or not showing the corresponding link(s).

b) *Adaptive Link Ordering*: these techniques sort the links of a given page, according to the user characteristics and adopting some criteria to emphasize some links from the others, e.g., the closer to the top of the page the link is displayed, the more important the target information is.

c) *Adaptive Link Annotation*: this kind of techniques allows complementing links with comments about the pages that are accessible through those links. These annotations can be textual (comments) or visual (icons, different colours or text fonts) and give information about the relevance of the target pages or their current state (visited, unavailable, etc.).

d) *Direct Guidance*: the goal of these techniques is to suggest the next best node, that is to say, the most relevant page to be accessed immediately after the current one.

e) *Adaptive Link Generation*: these techniques consist on the generation of new links, which have not been considered in the application authoring.

f) *Map Adaptation*: It corresponds to the distinct ways of adapting the navigational maps, influencing the structure or topology of the map.

In this section, we describe the way that OOWS allows describing adaptive navigation techniques at conceptual level. As a first step of this integration, we have focused

our work on the first three groups of techniques, privileging those ones that do not alter the topology of the navigational maps. For this reason, the consideration of *map adaptation* and *adaptive link generation* techniques are delayed to further works. In the same way, the consideration of *direct guidance* techniques is also postponed, because it requires a deeper analysis of navigational structures at instance level.

The adaptive techniques that this work proposes make use of the following three properties:

(a) *Accessibility*. It is a property applicable to Navigational Links and Navigational Relationships of a context. It allows constraining the access to the contents that are retrieved through these structures, depending on the user characteristics. This property is the basis of the implementation of Link-Hiding techniques for the mentioned structures.

(b) *Importance Order*. This property supports the Link-Ordering and Link-Annotation techniques for Navigational Relationships. For each context, a value is assigned by the modeller to each relationship, depending on the importance that he estimates each relationship has for a given stereotype.

(c) *Relevance.* It refers to the importance of the domain concept instances to the current application user. It is used by the OOWS adaptive techniques that are defined for navigational class instances. According to the adaptivity requirements, the developer describes the relevance in terms of the primitives included in the Domain-Dependent User Diagram. For instance, in the e-bookstore application, the relevance of a *B* book for a *C* client can be described as follows:

- The number of instances of *Keyword* class that are related to *B* and also with books that have been already purchased, owned, marked as favourite book, highly rated or visited by *C*. This definition is based on the following concepts (included in Domain-Dependent User Diagram, see Figure 8): *Book*, *Keyword, Client*, *Purchase* and *Ranking* classes; *Book-Keyword*, *owns*, *my_favourites* and *visited* relationships.
- The number of visited relationships between *C* and Book instances that are related to the same *Author* instance that is related to *B.* The considered concepts are: *Book, Author* and *Client* classes; *Book-Author* and *visited* relationships.

The adaptive techniques that this work introduces will be described according to their corresponding group of techniques (*link-hiding*, *link-ordering* and *link-annotation*), and to the conceptual structure to be adapted. Table 1 shows the type of techniques that this work comprises, along with the OOWS navigational structures to which those techniques are applied.

Table 1: OOWS Adaptive navigation techniques

	Navigational Class instances	Navigational Relationships	Navigational Links
Link-Hiding	X	X	X
Link-Ordering	X	X	
Link-Annotation	X	X	

4.1. Adaptive Link-Hiding.

OOWS provides the conceptual tools to define Link-Hiding techniques for different navigational structures.

4.1.1. Link-Hiding for Navigational Links. This technique is based on the "accessibility" property of this structure. Each navigational link of the Navigational Map that is defined for a given stereotype is marked with the "accessible" or "not accessible" value, according to the adaptivity requirements. If a link is "not accessible" to this stereotype, these users can not access to the target Exploration Context that is target of that link.

For instance, in the online bookstore, the developer has assigned accessibility values to the Navigational Links for users of *Client* and *User (non-client)* stereotypes. Figures 10 and 11 show the resulting Navigational Maps for each of these stereotypes, respectively.

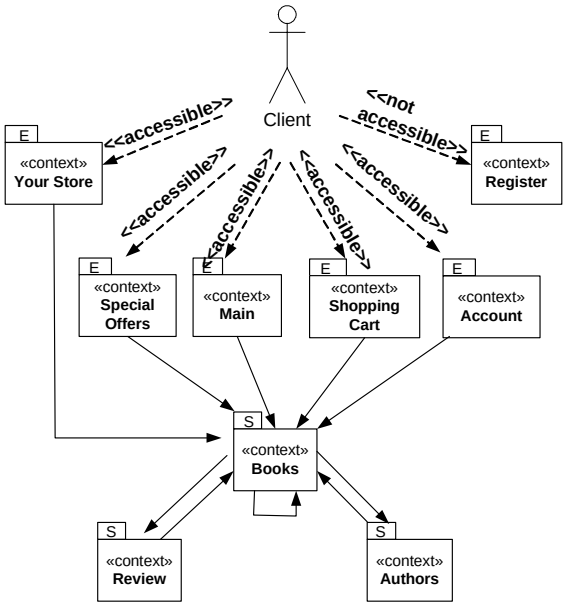


Figure 10. Accessibility values for navigational links for Client stereotype.

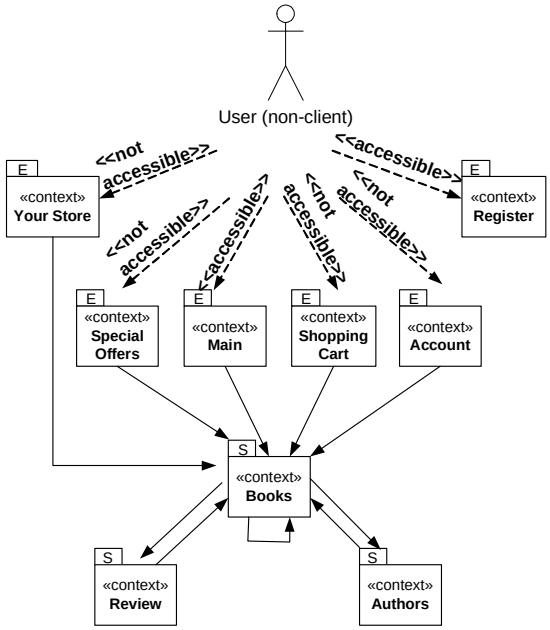


Figure 11. Accessibility values for navigational links for Client stereotype.

Considering the assigned accessibility values, a *Link-Hiding* technique is implemented, deactivating the <<not accessible>> links. Figure 12 shows the implementation for *Client* stereotype, in which the link to the Registration page is the only one that is inaccessible, whereas in Figure 13, corresponding to the implementation for *User non-client* stereotype, most of the displayed links are deactivated.

Figure 12. Link-Hiding for Client stereotype.

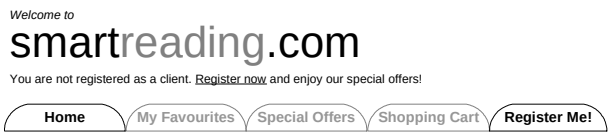


Figure 13. Link-Hiding for User (non-client) stereotype.

4.1.2 Link-Hiding for Navigational Relationships. Our proposal allows managing the access to Sequence Navigational Contexts by means of defining the Accessibility property for Navigational Relationships. For each context of a Navigational Map, the developer should assign the "accessible" or "not accessible" value to its navigational relationships, according to the different user. The accessibility property is defined for the navigational relationships whose source is the manager class of their context. The rest of the navigational relationships inherit the accessibility values from the relationships that allow accessing to them.

Figure 14 shows the accessibility values that the modeller has assigned to the navigational relationships of the *Book* context, considering the users of the *Infrequent Reader* stereotype. The user feature of not making any previous purchase on the site has been also considered, that is to say, in the Domain-Dependent User diagram there is no object of *Purchase* class that is associated to the correspondent instance of *Client* class.

In the example of the Figure 14, the modeller has considered that this kind of users has little interest on the *Specialized Review* of the book, marking the corresponding relationship with the "not accessible" value. Let us consider the following requirement of adaptivity: *"the access to a special offer associated with a book is restricted to those clients with a certain amount of previous purchases"*. According to the characteristics of the stereotype, the additional user feature compels to assign the same value for the *Book-Special Offer* relationship.

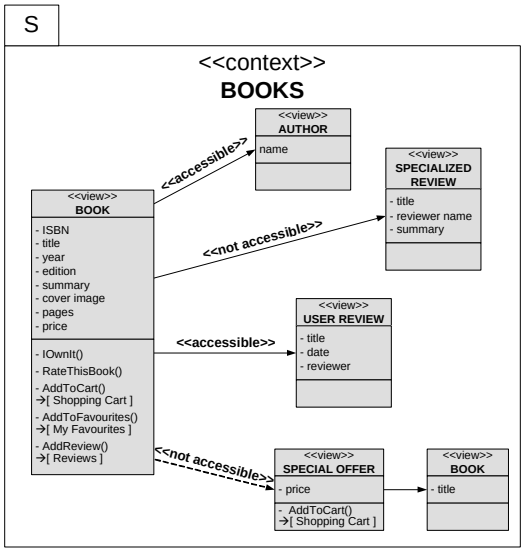


Figure 14. Accessibility Values assignment to the navigational relationships of Book context.

Figure 15 shows the implementation of the resulting context. We can see two different ways of hiding the links corresponding to the <<not accessible>> relationships. In both cases, the whole retrievable information has not been displayed. However, in the case of the *Book-Special Offer* relationship (see Frame 4), some information indicates the existence of information that is not accessible.

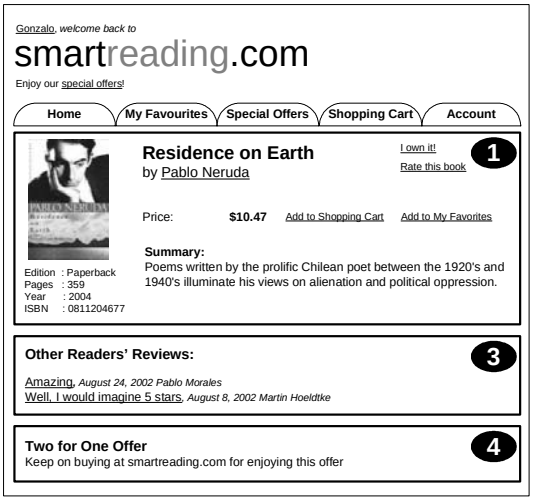


Figure 15. Implementation of Link-Hiding for Navigational Relationships.

4.1.3 Link-Hiding for Navigational Class Instances. This technique allows hiding the links to some instances of a navigational class, according to their corresponding values of relevance for a given user. This kind of link-hiding technique is applied to one or more navigational contexts whose manager class is the navigational class to be constrained. This manager class is extended with a filter, which constrains the objects to be shown. This filter allows accessing to the objects that have a relevance value greater than a given fix value, determined by the modeller.

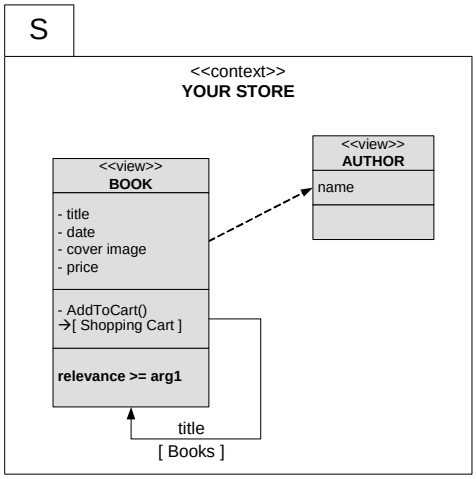


Figure 16. Conceptual specification of Link-Hiding technique for instances of the Book navigational class.

Figure 16 shows the applying of this technique to the context shown in Figures 4 and 5. The modeller constrains the books to be proposed, showing only those whose relevance for the current user is equal or greater than an *arg1* value. The final implementation of this technique is shown in Figure 17.

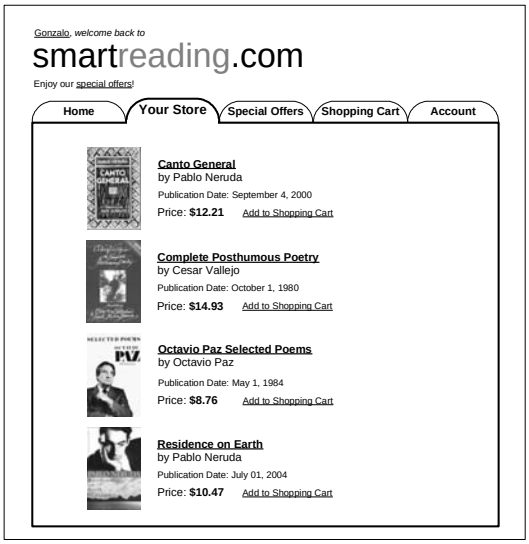


Figure 17. Implementation of Link-Hiding technique for instances of Book navigational class.

The context of Figure 16 only differs from the shown in Figure 4 on the inclusion of the filter by relevance. From a little change at conceptual level, it is possible to obtain an implementation highly different from the presented in Figure 5.

As Figure 17 shows, the links to the Book instances maintain their original order, but those ones whose relevance for this user is less than *arg1* have been hidden.

4.2. Adaptive Link-Ordering and Link-Annotation.

4.2.1 Link-Ordering and Link-Annotation for Navigational Relationships. Both kinds of adaptive techniques are supported by the Importance Order Property of Navigational Relationships. The modeller assigns importance values to the navigational relationships whose source is the manager class, establishing an order of importance among these relationships for a given user stereotype. In this way, displaying of information retrieved from one or another relationship is prioritized.

Figure 18 shows an example of importance values (depicted with the <<IO:value>> tag) that the modeller has assigned to the contextual relationships of Books context, for the *Occassional Reader* stereotype (*Client* instances with "*occasional*" value for the *readingHabit* attribute).

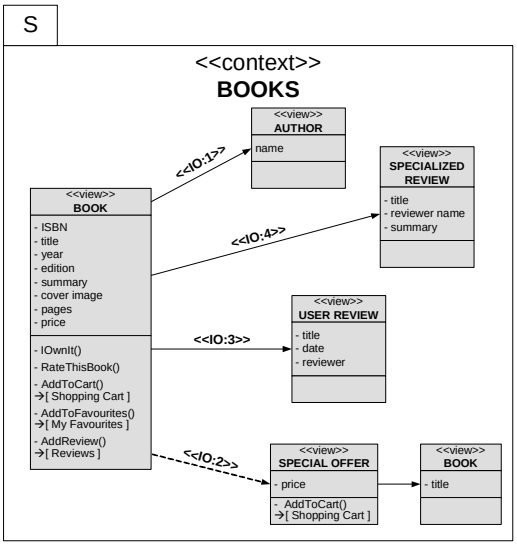


Figure 18. Importance Order values for Navigational Relationships.

The modeller assigns the initial order values, which may be modified according to adaptivity requirements. For instance, data about some user actions (modelled in the Navigational Behaviour diagram) may increase the score of the corresponding relationship for this user.

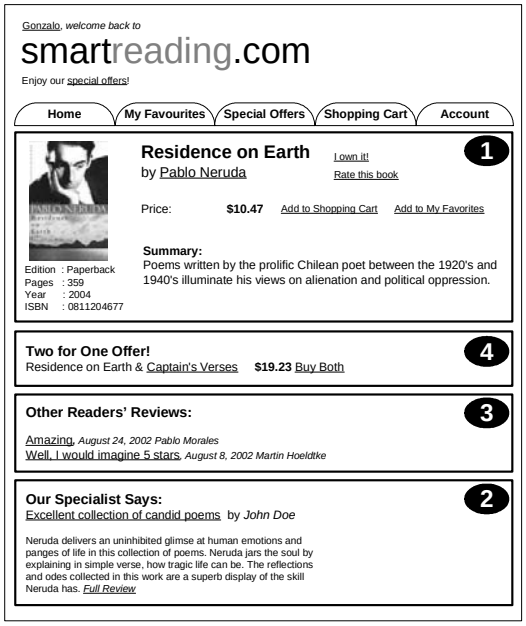


Figure 19. Implementation of Link-Ordering technique for Book Navigational Context.

The determined order of importance is used for the implementation of a Link-Ordering and a Link-Annotation technique. The difference between them is noticeable in the Presentation Layer of the application. In the first case (Figure 19), the information items that are retrieved through the navigational relationships are ordered in the layout of the page: items of the most important relationship are closer to the top of the page; in the case of the Link-Annotation (Figure 20), the information items maintain their original location into the page, but a visual hint is assigned to each of them, giving information of the individual importance for the current user (in Figure 17, a bigger quantity of "book" icons next to the data items means that the implied relationship is more important).

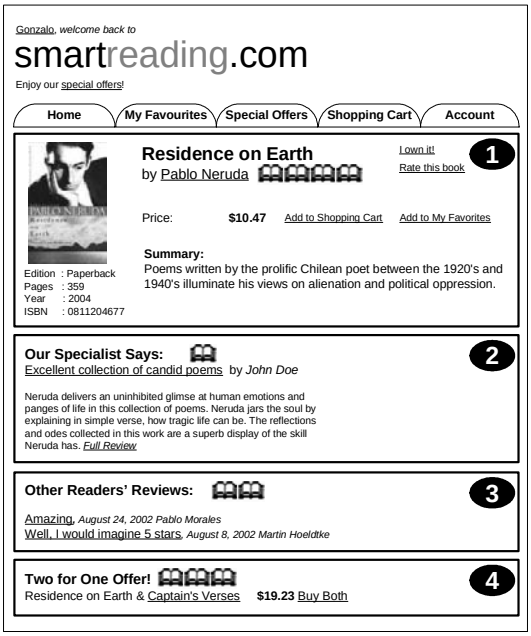


Figure 20. Implementation of Link-Annotation technique for Book Navigational Context.

4.2.2. Link-Ordering and Link-Annotation for Navigational Class instances. This technique allows ordering the links to the instances of a navigational class, according to their corresponding values of relevance for a given user.

Considering the navigational contexts whose manager class is the navigational class to be constrained, the link-ordering technique is described in the OOWS Navigational Model by means of the Ordering pattern. The developer incorporates this pattern to the manager class, ordering its instances by the relevance values, in descendant or ascendant order.

In Figure 21, the requirement of presenting the proposed books in a descendant order of relevance has been modelled. Figure 22 shows the resulting implementation. The whole information associated to the relevant link is ordered. On the top, the most relevant book to this user.

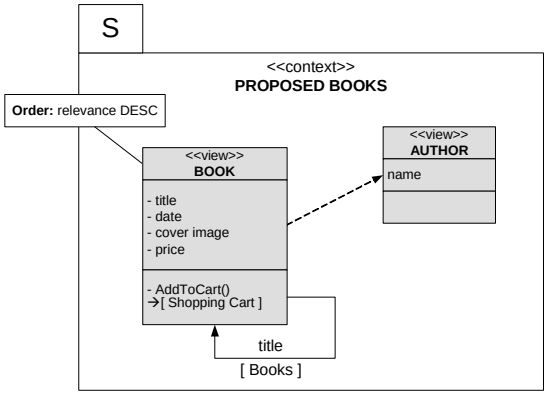


Figure 21. Conceptual specification of Link-Ordering technique for instances of Book navigational class.

There is not a navigational description for the Link-Annotation technique for Navigational class instances. In this case, the relevance order of the instances is considered directly by the presentation layer of the application, where visual clues are added to each link (more coloured "star" icons implies a more important book, see Figure 23).

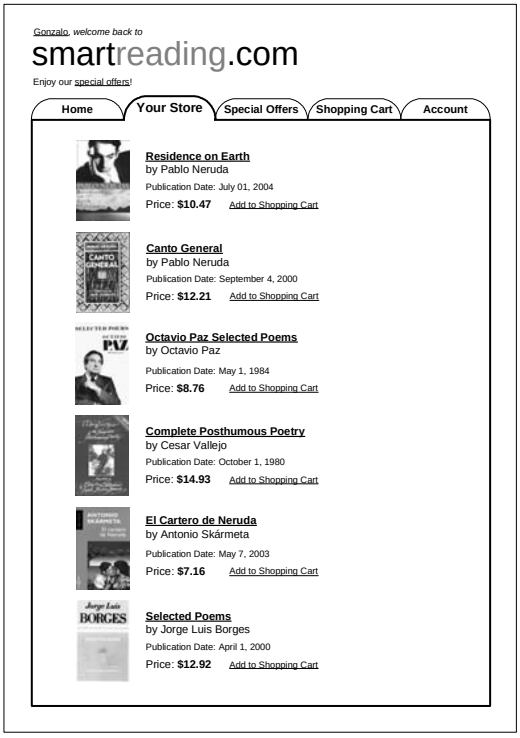


Figure 22. Implementation of the "Your Store" functionality, applying Link-Ordering technique.

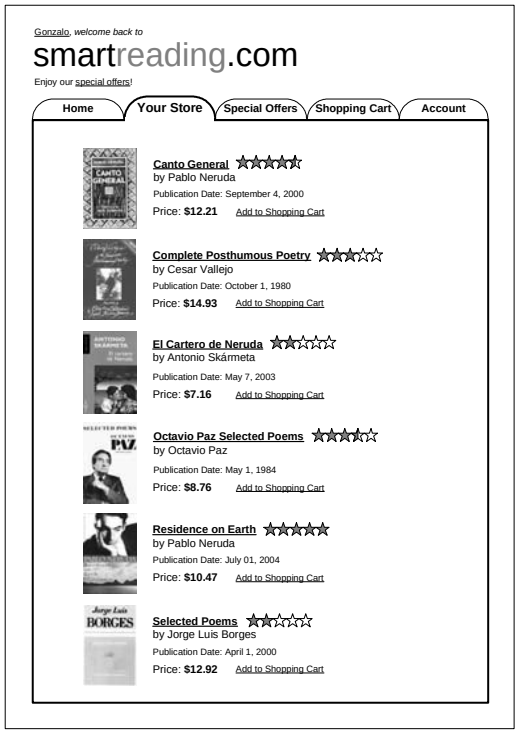


Figure 23. Implementation of the "Your Store" functionality, applying Link-Annotation.

5. Related work.

Some existing Model-driven Web development methods have been increasing the expressiveness of their models to support adaptivity features. Most of them are based on the definition of adaptive rules. For instance, in the *WebML* adaptivity approach [10], a set of Event-Condition-Action rules allows the system to make adaptations of its navigational structure in reaction to context changes produced by link activations. The adaptivity proposal of *OO-HDM* [3] describes users inside the structural model, assigning roles and defining algorithms that implement different adaptive rules for distinct user profiles. The *OO-H* proposal [4] separates the static and the variable parts of the Web application. By means of XML adaptive rules, the adaptation occurs over the variable part. Even if the OOWS proposal considers implicitly the definition of adaptive rules, it describes the adaptive features at a higher abstraction level in terms of adaptive techniques, instead of making rule-based descriptions. In this way, we obtain adaptive descriptions that are closer to the adaptivity requirements, intuitive for the modeller and with a more direct mapping to the final implementation.

Both *HERA* and *UWE* adaptivity approaches make a clear separation of the modelling and adaptation of content, navigation and presentation. This characteristic is also shared with our approach. The *HERA* approach [2] specifies the adaptivity by defining schema views on data for each model, performing adaptation without referring to concrete data instances. The main difference with *HERA* approach is that OOWS is object-oriented, providing a higher expressiveness in terms of the functional dimension of the Web application. The *UWE* approach defines a reference model [6] that formally specifies the features of an adaptive hypermedia application, incorporating a user meta-model and an adaptation meta-model. Our work can complement this proposal, providing the expressiveness to precisely define the concrete adaptive techniques to be implemented in terms of *UWE* adaptive rules.

6. Conclusions.

This work has presented our approach for incorporating adaptive navigation techniques into the conceptual modelling process for Web applications. We have augmented the expressiveness of a navigational model, introducing the required concepts to describe some well-known adaptive navigation techniques at a high abstraction level.

The flexibility of the OOWS navigational model makes possible that a non-adaptive specification can evolve into a diagram of an adaptive Web application, by using simple primitives. Both the User modelling proposal and the modelling of adaptive techniques have made use of concepts previously developed by Adaptive Hypermedia and User Modelling research communities. This help to increase the interoperability of the obtained Web applications with existing Adaptive Hypermedia Systems.

Some future works that complement this proposal are: (1) describing adaptive techniques of direct guidance, map adaptation and link-generation to the navigational modelling process; (2) describing adaptive presentation techniques, through the OOWS presentational model, adopting a similar approach; and (3) defining rules for updating the Domain-Dependent User diagrams from the navigational user actions.

7. Acknowledgments.

This work has been developed with the support of MEC under the project DESTINO TIN2004-03534 and cofinanced by FEDER.

10. References.

[1] A. Kobsa, "User Modeling: Recent Work, Prospects and Hazard", *Adaptive User Interfaces: Principles and Practice,* North Holland Elsevier, Amsterdam, 1993, pp. 111-128.

[2] F. Frasincar, G. J. Houben, and R. Vdovjak. "Specification framework for engineering adaptive web applications", *Proceedings of The Eleventh International World Wide Web Conference*, Honolulu, Hawaii, USA, May 7-11, 2002.

[3] G. Rossi, D. Schwabe, and R. Guimarães, "Designing Personalized Web Applications", *Proceedings of the World Wide Web Conference (WWW'10)*, Hong Kong, China, 2000, pp. 275-284.

[4] I. Garrigós, J. Gómez, and C. Cachero, "Modelling Dynamic Personalization in Web Applications", *Proceedings of Third International Conference on Web Engineering (ICWE'03)*,LNCS, Vol.2722, Springer-Verlag, Oviedo, Spain, 2003, pp. 472-475.

[5] J. Fons, V. Pelechano V., M. Albert, and O. Pastor, "Development of Web Applications from Web Enhanced Conceptual Schemas", *Proceedings of the International Conference on Conceptual Modelling, 22nd Edition, ER'03*, LNCS, Vol. 2813. Springer-Verlag, Chicago, USA, 2003, pp. 232-245.

[6] N. Koch, and M. Wirsing, "The Munich reference model for Adaptive Hypermedia Applications", *Proceedings of the Second International Conference on Adaptive Hypermedia and Adaptive Web Systems*, LNCS vol.2347, Málaga, Spain, 2002, pp. 213-222.

[7] O. Pastor, J. Gómez, E. Insfrán, and V. Pelechano, "The OO-Method Approach for Information Systems Modelling: From Object-Oriented Conceptual Modelling to Automated Programming", *Information Systems, Vol. 26, N. 7*, Springer-Verlag, Chicago, vol.2347, 2001, pp. 507-534.

[8] P. Brusilovsky, "Methods and Techniques of Adaptive Hypermedia", *User Modeling and User-Adapted Interaction*, *6 (2-3)*, 1996, pp.87-129.

[9] P. Brusilovsky, "Adaptive Hypermedia", *User Modeling and User Adapted Interaction, 11*, 2001, pp. 87-100.

[10] S. Ceri, P. Fraternali, and S. Paraboschi, "Data-driven, one-to-one web site generation for data-intensive applications", *Proceedings of 25th International Conference on Very Large Data Bases (VLDB'99)*, Morgan Kaufmann, Edinburgh, Scotland, September 7-10, 1999, pp. 615-626.

A UML Extension for Designing Usable User Experiences for Web Applications

Vito Perrone[1], Luca Mainetti[2], Paolo Paolini[1]

[1] HOC (Hypermedia Open Center), Politecnico di Milano, Italy
perrone@elet.polimi.it
[2] SET-Lab (Software Engineering and Telemedia),
Università degli Studi di Lecce, Italy
luca.mainetti@unile.it

Abstract

In this paper we introduce our framework for supporting the entire development of interaction and data intensive (typically Web) applications and describe one of the composing methods addressing the design of the user experience. Current proposals, both in the academic and industrial communities addressing such a kind of application, exhibit different weaknesses and strengths but are both characterized by poor acceptance by the current practice. Instead of proposing a new, richer modelling method, we have extracted and reused what good has been done in both the academic and industrial worlds in order to meet potential stakeholders' requirements. The whole approach has been shaped by the domain analysis and addresses the development of Web applications from requirements elicitation/analysis to software design in four phases. One of these phases, the user experience design named E-WOOD, is here detailed. Its specific stakeholders and requirements are here described. E-WOOD extends a UML proposal, coming from the industrial world, reusing web engineering principles coming from the academic experience. It introduces a reasoning oriented, user centered semantics which can be used for designing application better fitting stakeholders' goals and closer to final user expectations.

1. Introduction

Modern Web applications are assuming more and more a key role in most of the computer mediated human activities. ECommerce, eBanking, eFinancing, eLearning and so forth, are only some important fields where the success of the business is strictly related to the quality of the Web application acting as mediator towards final users. As far as shown in [1], in these applications user effectiveness and stakeholders' goals satisfaction are crucial for their quality. Typically, such applications provide users with a large amount of different information (data intensive applications) and integrate operations and business processes. As distinctive characteristics, all these services are accessed in a highly interactive way exploiting the navigational paradigm. Furthermore, currently services are more and more offered exploiting different delivery channels. These characteristics, together with the growing complexity of the delivered applications, have considerably increased the intrinsic complexity of such a software category.

Our assumption, quite agreed by the academic and industrial communities [2], is that quality for such applications is strictly related to a good design. As a consequence of the above considerations, it is clear that in this stage industry needs systematic design methods that could help in assuring the required quality [3].

Looking at the current methods addressing the design of Web applications, we discern two different communities. From the one side, we have the academic community, in particular the Web engineering one, where a considerable number of specific design methods have been proposed along the last decade [4-11]. Generally speaking and neglecting some peculiarities, most of these methods address the *conceptual design* [4] of Web applications. They are characterized by rich *semantics* coping with numerous peculiarities of Web applications. On the other side, there is the UML community where representative companies from the industrial world are employing a massive effort [17,18] in defining a reference modelling language for supporting the development process of software systems. UML native methods address the *logical design* [4] of an application, in that most of the modelling primitives abstract from concrete implementation artefacts. Being their modelling primitives closely related to concrete counterparts, these methods result easier to be understood and used by technicians as support for the implementation activities. On the contrary, when referring to the Web application domain, where the user interaction with the system plays a key role, it is recognized that UML support is quite vague [5]. It is mostly due to the fact UML lacks of a proper semantics which should help designers during the analysis activities to devise a user oriented application. Confirming this trend, the most recognized UML method proposed by the industrial community, that is, the WAE [12] introduce ah-hoc primitives for modelling typical software components of a Web applications, like *client page*, *server page*, *frameset*, etc. Evidently, these concepts do not belong to the user experiences so they cannot be used to reason on how to improve it.

Conversely, they can be effectively used to reason about the software architecture before coding it.

Besides their peculiar characteristics, the adopted modelling languages, primitives, strengths and weaknesses, both communities advocate the MDA as a way to improve the final product quality. However, they front this approach with a different philosophy. Academic proposals aim at designing what final users should perceive, carefully matching this design against the achieved requirements. Conversely, they often neglect architectural concerns. In other words, we can say they embrace the idea that quality is mostly decided at conceptual level so they typically pass from the conceptual design to the code. This position is also evident by the fact that most of these approaches propose support tools that can produce an application prototype (usually an evolutionary one) from the conceptual design. On the other hand, industrial approaches pay more attention to the correct design of the software modules that compose the system architecture often overcoming an accurate design of the user perspective. Most of them analyze user requirements by means of use case diagrams [18] describing which functionalities are supposed to be provided by the system to its users. Then, the user interaction with the system is detailed by means of sequences or collaboration diagrams [18] that describe the dynamic properties of these functionalities in some relevant scenario. On the basis of such an analysis and the chosen architecture, designers have to define the software that will meet the achieved requirements. In other words, we can say they embrace the idea that *quality can be assured by a good analysis and if suitable software models are crafted*.

Recent studies [3, 14, 15, 16] demonstrate that in the Web domain most of the current proposals have only slightly impacted on the actual practice. What are the reasons? Which of the above philosophies should be adopted to define a more suitable design model? Various can be the factors that hinder the adoption of systematic approaches for modeling.

In this light, the methodological framework we introduce in this paper aims at embodying the advantages of the two mentioned philosophies. It is composed by four phases embracing the web application development lifecycle from analysis to software specification. Each phase adopts a specific design model which has been defined on the basis of an accurate analysis of its stakeholders' goals. In particular, in this paper we describe our proposal for designing the user experience. This method, named E-WOOD, extends the Conallen's UML proposal for designing the user experience – UX [12] but embodies the semantics of a known academic method, called W2000 [25, 26], we are familiar to.

To illustrate vividly the approach and in particular the user experience modelling method, we will use real examples from the design of a running Web application which we have designed and developed: the Website "Munch und Berlin" (www.munchundberlin.org). It has been originally realized for the State Museum in Berlin within the HELP EU-funded project with the aim of providing to the general public (including users with visual disabilities) a Website promoting the temporary exhibition of Evard Munch's prints. Being a real, even if relatively small-sized, application, it is quite suitable to illustrate the main aspects of our approach.

2. Web application design: panorama and related works

Along the last ten years a number of methods have been proposed for supporting the design of Web applications. In the following of this section we briefly resume the main characteristics of these methods from two perspectives – the academia and the industry – and considering their role with respect to the analysis and software design phases. Looking at the academic community, some of the most known existing methodologies are HDM [6], W2000 [25, 26], OO-HDM [8], WebML [10], UWE [11], WSDM [9], OO-H [13], etc. Roughly speaking, they specify the design of a Web application at the conceptual level, neglecting technological aspects and constraints. Besides technical (minor) differences, these methods share lots of common features. All of them are based upon an information-navigation paradigm to describe the user interaction, recognize the importance of the semantics as guidance for conceiving the application design and share the fundamental principle of *separation of concerns*. On the other hand, they differ one with another in terms of proposed design primitives, notation and support tolls. All these methods Following this principle, and adopting the W2000 [25, 26] terminology[1], the design of a Web application is achieved in four dimensions: *Information and Access Structures design*, defining the basic conceptual information units (entities) as perceived by the user and the navigational infrastructure in terms of semantics associations (between related entities) and

[1] In this paper we use W2000 as example of academic design method since it has been developed in our research group so we are very familiar to its terminology. Nevertheless, we are firmly convinced all our considerations are quite independent from it and generic with the respect of other similar design methods.

access structures (navigational paths enabling users reach interesting information units); *Operations and Business Process design*, defining operations (e.g. "add to shopping cart") and processes (e.g. "check-out", "registration") within a Web application; *Navigation design*, defining the navigation network allowing users browse information and access structures and execute operations and processes; *Presentation design*, defining the page structure in terms of lay-out aspects and graphical elements and the page organization and navigation.

Although, if properly used, current academic methods have the potentiality of enabling designers conceive high quality (say usable and effective) applications, they suffer, as stated in a recent study [3], of some inefficiencies which contribute to a poor acceptance from the industrial environment. Owning sophisticated and semantically rich primitives often it *takes too much effort and time in order to learn and start using the methods*. *Modelling purpose is only badly or vaguely specified* with the respect of the overall development process. It is often claimed models are intended as support tool during the early analysis activities, but they then their models are also used to automatically generate the running application [13], [10]. *Cumbersome design documents* are generally produced as output of the design activities. These documents risk being hard to read and use both during the analysis and the following implementation activities. *Proprietary concepts and notation* are generally proposed (except a few cases like [11]) by each method increasing the learning time and thus the negative perception of industry people [21]. *Ad-hoc and in-house made support tools* are generally proposed instead of commercial ones.

With regard to the second category, that is, methods proposed in by the industrial world, UML is definitively considered the standard de-facto in the design practice. Referring to the web application domain, the only recognized method coming from the industrial environment is the one proposed by Conallen in [12],[20], that is, the Web Application Extension (WAE). WAE, like other UML native methods, adopts an implementation oriented approach, in that most of the modelling primitives abstract from concrete implementation artefacts. Examples of WAE primitives are *client page*, *sever page*, *style sheet*, *frameset*, etc, obtained by stereotyping UML classes and *link*, *redirect*, *submit*, etc., obtained stereotyping UML associations. Due to this characteristic, they are quite easy to understand and use by technicians for supporting the software design activities and broadly supported by commercial tools. On the other hand, concerning WEB applications, it is known [5] that UML lacks of proper semantics for supporting the design of communication and navigation aspects both during the analysis and design phases.

Finally, the topic of explicitly considering stakeholders and their requirements for shaping a suitable design method has been barely fronted by existing approaches. In most of examined literature when a new modelling method is proposed, the well-known and high level software engineering principles are, at most, cited. For example in [5] it is argued that the next generation of OO methods "*…should be sufficiently user-friendly to all kinds of possible stakeholders. That is, for all stakeholders of any model, its relevant parts expressed in the modeling language, must be understandable, must be clear even. For the modeler as well as for all other persons involved in the modeling activity, any model must be expressive, precise and clear as well*". However, besides these well known software engineering principles, we also advocate that, due to the diversity of all possible stakeholders, the lack of an explicit consideration of what every potential stakeholder expects by the modeling method could be one of the main reasons of the existing gap between current proposals and industry practice.

3. Analyzing Stakeholders' Requirements

To be successful, design methods, as well as any engineering product, should accomplish the needs and expectations of its potential stakeholders. Defining a new method requires an accurate analysis of goals and requirements of their users, i.e. the practitioners who daily conceive, develop and deploy applications, and other potential stakeholders whose needs may influence the method definition. Neglecting stakeholders' needs can bring to lack of attention towards these engineering products (design models) by the industrial practice while fitness to requirements can drastically increase their acceptability at wider level. On this basis, we have defined our approach by taking explicitly into account its potential stakeholders. It is composed by four phases embracing the web application development lifecycle from analysis to software specification. Each phase adopts a specific design model which has been defined on the basis of an accurate analysis of its stakeholders' goals. In this paper we focus on the conceptual design of the user experience which is usually achieved between the analysis activities and the software design.

3.1 Requirements for a conceptual model addressing the user experience design

Conceptual models are used at the beginning of the overall design activities, as intended in the software engineering discipline, which will finally lead to the detailed specification of the software modules to be

coded. In this phase, the main goal of conceptual models is to clearly define the solution (application to-be) characteristics, even if still avoiding implementation details. In the following potential stakeholders (the most relevant ones) and their relative requirements, gathered in our experience on the field, are described. It should be noticed that not all the described stakeholders are also active users of design method, but their needs can indirectly influence the method definition.

Designers: are in charge of the system design. Depending on the reference community, the terminology adopted within a company, the kind of application, and so on, different professional figures (e.g. information architects, interaction and usability experts, and so on) might be attributed to play this role. Usually several designers work both in the analysis and design phases thus first goal is to ***ease the communication with the analysis activities and among different designers in the design activities***. For the former, some form of *guidance* should be provided *to support the passage from the early solution devised in the analysis activities to the actual design* of the system. This *mapping should compromise between rigour* – to enable some form of automatic passage – *and flexibility* – to not constraint choices designers have to perform in the design phase. In this phase, they have to design models very close to the application to-be, thus inevitably these models are rich in details and the specification is often composed by several heterogeneous diagrams representing different application concerns. To master the overall design complexity (avoiding naive designers feel lost) the method should *provide an explicit framing strategy*. Furthermore, model drawing is time-consuming activity that needs proper tool support. In order to be used in professional environments, *support tools should adhere to the commercial standards*. Since building such tools is an expensive activity, new modeling methods should be defined so that *existing commercial tools can be exploited*.

Usability experts and Graphical designers*:* depending on project parameters like those mentioned above, these roles could be attributed to designers or other professionals with non technical skills. However, in WEB applications these aspects are taking more and more importance and require specific competences. Whatever is the case, these figures are interested in carefully ***defining and reviewing usability and graphical aspects*** of the application to-be, thus *concerns impacting usability and layout/graphical aspects should be explicitly modeled and made easy to access*. These experts are used to analyze and discuss about usability and graphical concerns by means of mock-up or other similar representations that closely reproduce the application to be. Thus, to achieve an effective communication with usability and graphical experts, *models should* also *look as close as possible to the actual application*.

Software designers and Implementers*:* define and implement the software modules that will actually realize, on the basis of the chosen system architecture, the application specified by the conceptual models. From our experience on the field, a recognized lack of ***existing conceptual models*** is that they ***require a considerable effort to be mapped into software artifacts***. Often, it is hard to understand which diagrams should be considered for obtaining a single software artifact and, most of times, several different diagrams must be composed. For example in the web domain, to design a server page, software designers have to refer to information models for the page data, operation and business process models for the business logic, navigation models for the navigation logic and presentation models for graphical and layout aspects. Software designers consider this activity being time consuming and, if not properly supported by tools, a possible source of mapping mistakes. On the basis of these considerations, models should *embody modeling primitives as closer as possible to concrete counterparts* and that *as less as possible diagrams should be considered to define a software component*. Also the *design documentation to be used for supporting the implementation activities should be concise and easy to read* (many cross-references among different diagrams are considered highly annoying). Another highly desirable feature a modeling method should own, for these stakeholder types, is to *provide predefined mapping strategies – let's say mapping patterns – towards the most known architectural patterns*. Finally, most of the interviewed software designers and implementers were already used to the UML and related CASE tools, thus they showed a remarkable preference in having conceptual models described in UML-like notation and following the UML philosophy, that is, modeling methods should *belong to the UML family.*

Product manager: this stakeholder type represents the most important client counterpart dealing with the application design, and act as interface of decision makers, opinion makers, clients and content/domain experts. Product managers are usually in charge of assuring the envisioned application will be able to satisfy the client company expectations, but they also are responsible of a number of other specific tasks. Among others, one the most important is to set up the editorial chain. Their main, somehow opposite, goals are *to take the control of the overall application at a glance* and *to get details of specific aspects (related to their tasks).* Desirable features for the method should be to *review models at different levels of detail*, to *embody most of the needed information to set up the editorial chain* and to *enable some form of requirements tracking*.

Final Users*:* this stakeholder category is the more important for tuning the application interaction even if it is also the less accessible for several reasons. In fact, they

usually are not part of the client, are barely identifiable and their characteristics can vary remarkably. Nevertheless, gathering some feedback from potential users before the coding activities start can bring several advantages since modifying models is much less expensive than modifying code. From our experience [27], a discussion with users mediated by models is usually ineffective because they need to see and handle application as it were running. Application prototypes are much more effective in this development stage, thus *models should be easy to turn into prototypes.*

Testers and Evaluators*:* models produced in the design phase are also used by testers and evaluators once the application has been implemented. In these phases, models should provide the ground for setting up the testing or evaluation plan. Testers and evaluators need *different concerns to be evaluated being easily identifiable* in the implemented application. Moreover, *models should look very close to the implemented application* so that testers and evaluators can easily match the running product to the originating models.

Table 1. Requirements for a conceptual tool for the design phase.

Stakeholders	***Design Requirements***
Designers	*R1. provide guidance for passing from early solutions to actual design* *R2. compromise between rigor and flexibility* *R3. provide a framing strategy* *R4. enable to exploit existing commercial tools*
Usability and Graphical experts	*R5. distinguish and make easily accessible concerns impacting usability and layout/graphic* *R6. models should look as close as possible to the actual application*
Software designers and Implementers	*R7. modeling primitives as closer as possible to concrete counterparts* *R8. as less as possible diagrams should be considered to define a software component* *R9. Concise and easy to read specification documents* *R10. predefined mapping strategies towards the most known architectural patterns* *R11. belong to the UML family*
Product manager	*R12. review models at different levels of detail* *R13. embody information to set up the editorial chain* *R14. enable requirements tracking*
Final Users	*R15. models easy to turn into prototypes*
Testers or Evaluators	*R16. different concerns to be evaluated being easily identifiable* *R17. models should look very close to the implemented application*

4. The whole framework at a glance

In this section we briefly introduce the whole methodological framework to better contextualize the proposed conceptual modelling method. In all the section, we specify precise references to the requirements discussed above as it becomes necessary.

In Figure 1 the composing phases are shown. A different modelling method is proposed for each of them. As well as other software development processes, we assume that these phases should be executed in an iterative and incremental way, therefore the picture only purpose is to express the phases order within the whole process. Considering the entire development process of a web application, we can say the framework covers both the *analysis* and *design activities* [28]. Moreover, adopting the Jackson terminology [22], we distinguish between the *problem* and the *solution domains*. These dimensions, the process and the domain, are used to organize the following discussion.

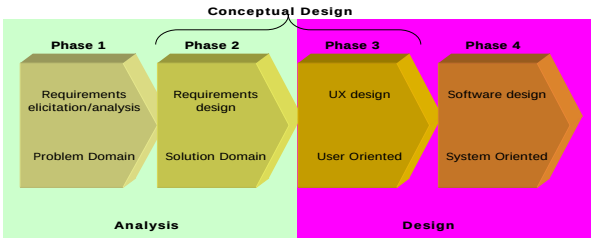


Figure 1: Phases in the development process of Web applications

The two left more phases are both achieved during the analysis activities. For supporting the **requirements elicitation and analysis** (phase 1) we propose AWARE [1], a goal-oriented method specially suited for web application requirements engineering. AWARE primitives include *goals* and *requirements* which definitively belong to the problem domain.

However, in our experience, discussing with stakeholders (analysis) about needs and goals can be too abstract for a fruitful reasoning about relative importance of various goals and requirements and for eliciting new ones [27]. A first very high level solution, focusing on specific topics, can help validation and elicitation activities (e.g. interviews) enabling a more concrete discussion about the problem. We call this activity **Requirements Design** (phase 2) meaning that in this phase requirements take a more concrete form accomplishing a preliminary hop from the problem domain to the solution one. In this phase we use IDM [24]. IDM (Interactive Dialogue Model) is a design model for interactive applications based on linguistic concepts of human dialogue. It bases on the interpretation of the interaction between the user and the application as a sort

of dialogue. It is simple to grasp, and effective in representing the most relevant features of the application in terms of content of the dialogue and dialogue moves. In fact, three simple design elements characterize IDM: “topic”, “change of topic”, and “group of topic”. An interactive application may describe a “topic” (e.g. a “print”, or a “technique”); or it may allow the user to switch to a “related topic” (e.g. switching from a “print” to the “technique” used for it); or it may allow the user to start from a “group of topics” (e.g. “the masterpieces”, or “the prints dealing with sickness”) and then browse within the group.

Although in traditional SE approaches requirements are directly used for designing the software architecture (e.g. class diagrams, component diagrams, etc. using the UML terminology), in applications where the user interaction and the communication potential play crucial roles, the software design has to be postponed to the *user experience* design [12]. In this phase the application is designed as perceived by final users, neglecting how the software will be realized. Here, designers have to precisely define how users interact with the application to accomplish their tasks, taking care of the application usability and effectiveness with the respect of user requirements and quality expectations. In our framework, we achieve the concrete passage into the design phase by translating IDM models (phase 2) into E-WOOD ones (phase 3). IDM and E-WOOD, together, build up our approach to the conceptual design of WEB applications. Both methods take their foundations in W2000 [25, 26], last heir of HDM [6] recognized as one of the first conceptual methods for web application design. As described in section 2, W2000, as well as other similar conceptual models, implements the *separation of concerns* principle by structuring the design in four dimensions. Both our methods keep this principle at the basis of their definitions but projecting the previous dimensions in a sole dimension for the sake of conciseness, for reducing the number of concepts to be learnt and references among diagrams (R8,R9). The last step (phase 4) consists of a detailed design of the software that will be implemented to *realize* the desired user experience. This is generally called *logical design* of the system to-be. Passing from phase 3 to phase 4, a paradigm shift is achieved since, in phase 4, designers have to design the *system* that will *realize* the modelled user experiences. This passage is far to be straightforward and a number of trade-offs with the architectural constraints and various decisions have to be undertaken [27]. Models produced in this phase should specify a design easy to code. Here, we adopt the modelling method proposed by Conallen, namely WAE [12]. Our choice has been driven by two main reasons. First, it is already recognized in the industrial environment as the UML method for designing the software for web applications and a number of CASE tools already support its diagram drawing (e.g. Rational Rose, MS Visio). Second, as shown in paragraph 4.2, it is very easy and intuitive mapping WAE models upon E-WOOD as far as most of times, only one E-WOOD artefact is needed to define a set of related WAE artefacts (R8,R9).

Finally, the methodological framework also includes a number of guidelines on how to use every method within each phase and how to move forward and back between adjoining phases. Guidelines are informally described in terms of patterns [29] so providing an useful but flexible guidance (R1,R2). They also front specific design issues like the multi-user and multi-channel design. Lack of space prevents us to describe this aspect, but the complete set of guidelines can be found in [27].

5. E-WOOD: the user experience design

Our proposal for designing the user experience, called E-WOOD, has been defined as a UML extension. UML has been chosen as modelling language to meet R11, while the extension mechanism has been preferred to defining a metamodel in order to exploit easily existing commercial tools (R4). Our model extends an existing proposal for designing the user experience, that is, the UX [12] since, as shown in the Conallen’s book, mapping WAE models upon UX ones is easy and intuitive (R8,R10,R17). UX’s high level primitives are *screen* and *links*, and an application is merely considered as made up of a number of screens connected by links. Typically, a set of WAE artefacts are mapped upon a screen by means of *realization* associations (stereotyped as <<build>>), specifying which logical elements (WAE models) build the various parts of the screen (contents and links). Our main goal in extending the UX has been to add the needed semantics (extracted by the W2000 primitives) to enable the separation of concerns impacting the application usability, its functionalities and the whole quality (R5,R13,R16). In E-WOOD different concerns are specified in different views and by introducing specific design concepts. These concepts have been defined extending standard *class* and *association* elements in terms of *stereotype, semantic description, constraints, tags* properties. An additional property (*mapping constraints*) has been also introduced to specify mapping constraints between IDM and E-WOOD models (R1,R2). As well as in UX, E-WOOD high level primitives are *screens* and *links*. Screens can aggregate both content and input forms; links can be used to perform a simple navigation among pages or to provide inputs to operations and processes. E-WOOD models are thus very close to the application to-be (R6, R7, R17) and easy to turn into prototypes or mock-ups (R15). Keeping these basic primitives we have also preserved the proven mapping capabilities towards the WAE (R10, R8).

The introduced semantics is also used to define a framing strategy (R3) which helps designers organize the overall design activities, fosters reuse and make design documentation more readable (R9). The framing strategy mostly reflects the W2000's design dimensions. E-WOOD proposes to organize the design of the overall application in five views. Each view includes several diagrams and makes use of specific stereotyped classes. Due to the lack of space, in the following we only introduce the main views to show the philosophy behind our method and how we have tried to accomplish the above stated requirements. The complete specification can be found in [27].

The *Template View* is used to define common contents and links of page sets. Examples of common contents could be the copyright information, the company logo and so on, whilst examples of common links could be those connecting to the home page or to the various site's sections (like those on the bottom of many web sites). Typically the template design involves the graphical designers who are in charge of the application look-and-fell (R5). The basic primitive used in these diagrams is the <<Screen Template>>, an abstract class used as place-holder for content and links belonging to a set of screens. Layout contents (both information and graphical elements) and common links are modelled respectively by means of <<Layout Content>> and <<Landmark link>> primitives. In Figure 2 a Web page of Munch is shown together with design excerpts taken from the template view.

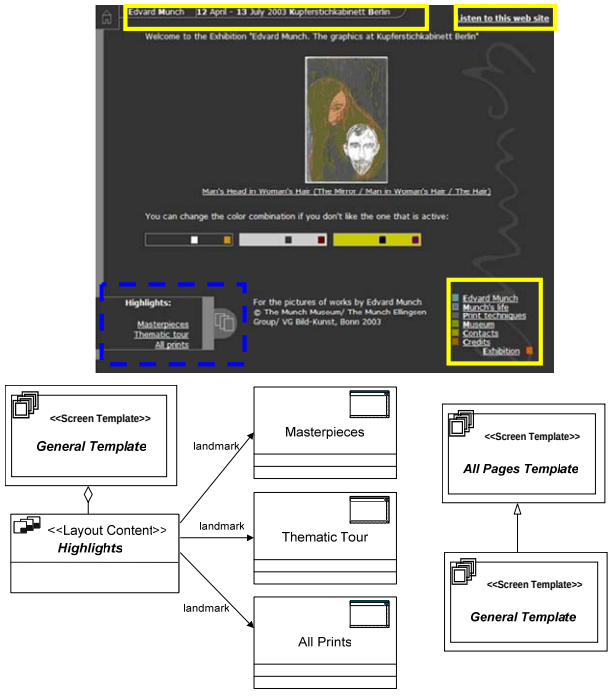


Figure 2: Some excerpts from the Template view

Every Munch's Web page includes contents and links highlighted in the picture by continue line rectangles, while only some pages include also the set of links highlighted by a broken line rectangle. To model this we use two <<*Screen Template*>> abstract classes modelling the two different templates. The diagram on the bottom right corner shows that a specialization relationship is used to specify the hierarchy between these templates. The layout content belonging to the "General Template" is represented by the <<Layout Content>> class aggregated to the template, while the outgoing links are represented by means of the stereotyped associations <<Landmark>> ending on the target pages. It can be noticed as <<Screen Template>> is modelled as abstract class since it is defined only for generalizing content and links belonging to a set of concrete pages. A specific constraint is provided in the formal specification of the UML extension.

The *Structural View* is used to define pages enabling users explore information concerning the domain entities or IDM' topics. <<Content>> classes are aggregated to screen classes and models portions of the whole topic information. <<Structural link>>s are used to model the navigation achieved across pages belonging to the same topic. For example, as depicted in **Figure 3**, the overall information concerning the "Print" entity are organized in three pages (Introduction, Big Image and Description) which are connected by means of bi-directional links originating from the "Introduction" page. Each IDM topic is mapped on a number of content classes (and relative pages) equivalent to the number of its dialog acts. Content classes are then enriched by a fine-grain definition of data slots which can be used as input for setting up the editorial chain (R13). Content classes contain a Boolean tagged value called *entry point* whose purpose is to specify whether that portion of the content can be used as starting point for exploring the entity information. Following our framework guidelines, such pages should include, at least, a minimal set of entity attributes that can be used by the user to understand what the entity instance talks about. Information organization, kind of navigation and entry points are concerns usually discussed with communication and usability experts (R5,R16) taking in mind that when users navigate these pages are clearly interested in improving their knowledge about the entity.

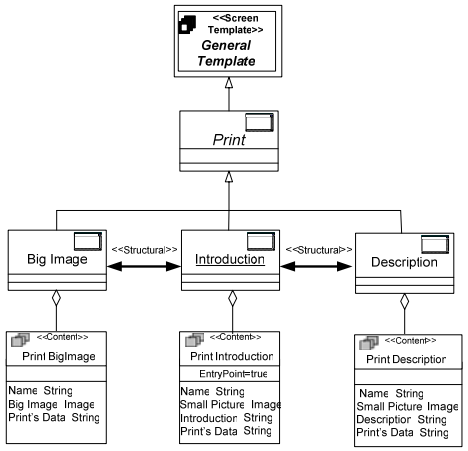


Figure 3: a) Structural view for the "Print" topic;

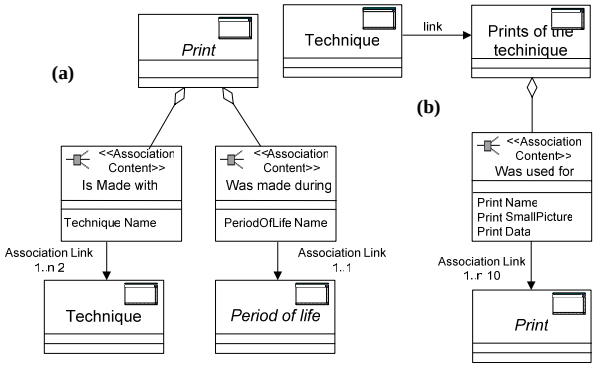


Figure 4: Association views

In the *Association View* designers specify how to pass from a discovered interesting topic to a related one (*relevant relation* in IDM). For this part of the user experience design, it is very important to carefully decide how to enable users to understand which of the possible target topic instances they are interested in. This aspect is called, in the HCI community, *information scent* and is one of the factors strongly impacting the application usability. In E-WOOD we use to this purpose the <<Association Content>> (**Figure 4** (a) and (b)). In (a) these information are integrated in all the "Print"'s pages (it is aggregated to the abstract page representing the common features of all the structural pages) and <<Association link>>s connect these pages to the target one. In (b), the "Technique" page includes a <<Link>> association which brings to new page "Prints of the Technique" whose only purpose is to list the possible target "Prints" which have been produced using the source "Technique". From this page a <<Association link>> point to the destination pages. The <<Association link>> primitive includes a tagged value that specifies the association *multiplicity* in terms of *min*, *max* and *expected* values. In particular, the expected multiplicity provides a useful indication about how many instances of the target entity are in general addressed by the association. This information can be used for taking some design choices like attaching the <<Association content>> to the source page or defining a new ad-hoc page (the two possible solutions shown above). Having *max* or *expected* cardinality very small, our guidelines suggest aggregating the <<Association Content>> to the source pages, while in case the expected number grows up, we suggest the other solution.

In the *Access Path View* designers have to specify the navigational paths enabling users find interesting objects. For example in an e-commerce web site like Amazon, examples of predefined navigational paths are the books categories which organize books by topic and sub-topics, but also a "Bestsellers" or personalized "Book recommendations" and so forth. The purpose of these pages is supporting users while exploring the proposed site content organization improving the user understanding. Such pages should help users in deciding how to move around possible choices enabling them exploring in depth the navigational path. In each path step, possible users should choice how to refine the set of possible interesting kinds of topics or, in case of terminal steps, which topic instance is worth to be examined (passing to the structural navigation) among the possible ones. Navigational paths are related to the IDM "Group of topics" concept. In the example in **Figure 5**, the E-WOOD model of a one-step path enabling users access to the most famous Munch's prints is depicted. The access structure is available in the page modelled by the <<Screen>> "MasterPieces". Here users can find an "Introduction" to the collection and a list of prints. For each print a short preview is provided by means of three print's attributes: "Small picture", "Name" and "Print's data". This information is modelled by the <<Access Content>> class aggregated to the <<Screen>> "MasterPieces". By means of these previews users can choice which print they are interested in and navigate to it by means of the <<Collection Link>> "Index". Once users land to the choose print page, he can also move back and forward among the collection members (other MasterPieces prints). To model this, in this diagram the <<Collection Link>> "Next/Previous Masterpieces" is added to the abstract <<Screen>> "Print". It represents

the E-WOOD model for a case of *guided tour* pattern. It is important to be noticed as these links are only available in the context of this collection, so if the user reaches a print by other access mechanisms (other access paths, associations, search engine, etc.) he cannot move among prints contained in this collection. Access paths usually define a *navigational context*, in that new content and links can be added to entity pages when accessed by a specific access path.

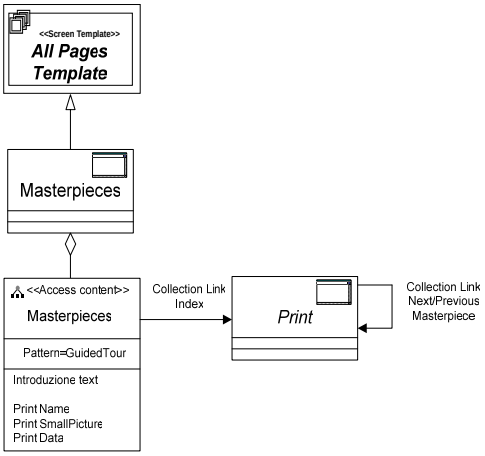


Figure 5: An excerpt of the Access Path View for the Masterpieces page

Besides these main views, we also propose a *Navigational Map View* that summarizes the main navigational features of the entire application. Our guidelines suggest how to choice candidate pages, among the overall defined in other views, to be included in the navigational map. Switching from the navigational map to the detailed design of contained screens it allows R12 being accomplished. Finally, the *Operation/Process View* complements the previous views adding to pages concerns related to operations invocation (e.g. "add to shopping cart") and defining pages involved in business processes execution. Lack of space prevents us to describe this complementary view.

6. Conclusions

We all know that the existing literature about conceptual methods addressing the design of Web applications is (over)abundant. On the other hand, we also know that a remarkable gap between theory and practice still exists [14, 15, 16],[3]. *Which are the reasons behind the poor acceptance by the practitioners*? Starting from these considerations, in this paper we have claimed that a possible reason might be that existing proposals have failed short by neglecting stakeholders' goals and expectations. In this light and focusing on the development of WEB applications, we have carefully analyzed the environment where a design method should operate identifying which are the potential stakeholder types and their goals and requirements. On the basis of this analysis, instead of inventing new design methods, we have reused or extended the best, in our view, of current approaches both in the academic and industrial communities. The approach covers both analysis and design activities and consists of four phases, executed in an iterative and incremental way. Defining it, we put in practice most of our experience achieved working on the field with conceptual design methods for Web applications [3], [27].

This paper has focused on our proposal for the user experience design, namely E-EWOOD. It is a UML profile that enables to specify the user experience in terms of pages and links but that embodies semantics enabling designers reason together with different stakeholder types about crucial concerns heavily impacting the application usability and effectiveness, that is, its perceived quality. Moreover, due to its definition, E-WOOD can exploit existing commercial tools for supporting the model drawing and perfectly match an existing and already affirmed, among practitioners, method for designing the software modules of a WEB application.

The approach has been applied in several design and reverse design case studies and industrial projects. Its transferability in industrial environments has been also experimented in two projects in cooperation with two Italian software companies (in the context of the GENESIS-D projects [27]). From these first experiences a number of considerations can be drawn out. Compared to W2000, we have noticed a significant decrease of the required learning time. Practitioners were able to use both methods after a short but intensive course (2-3 days). They drew IDM models using paper and pencil, while used VISIO™ stencils for designing E-WOOD and WAE models. In all the achieved experiences, we spent, with E-WOOD, on the average one third of the time required by W2000 to produce the same level of detail in the specification of several application designs. This has to be summed to the time required for manually drawing IDM models which is, however, very affordable. Compared to UX, we obtained several advantages mostly due to the introduced semantics. Models are more expressive and easy to be revisited; the framing strategy enables a suitable organization of the overall design activities; a number of well know design patterns, developed in the web engineering community, can be exploited to produce quality applications.

Finally, concerning future works, we are working in two main directions: (i) enriching the framework with guidelines and patterns for fronting specific aspects like the multi-channel design and the mapping of E-WOOD models upon the most known software architectures (JAVA and MS.NET); (ii) Concerning the second point, as said above, one of the reasons which guided our

decisions in defining methods in phase 3 and in adopting WAE in phase 4 has been the reuse of commercial CASE tools already widespread in the industrial environment like MS Visio® and IBM Rational Rose®. So doing, companies already accustomed to these tools can easily step towards the adoption of our methods only adding a few stereotypes and our views strategy. Existing commercial tools do not support the design of models of phase 1 and 2. In order to improve the coverage of the whole approach with proper tools, we are already working for defining an ECLIPSE [30] add-in which should include, besides all the modelling primitives, also a set of semi-automatic rules for passing from a phase to the next one, some tracking mechanism among phases and a loose consistency check option. For example, the consistency manager could check if the several user views can be actually derived by the unique database and if the logic perceived by users when executing processes is compatible with the business processes implemented at the logic level.

10. References

1. Bolchini, D., Paolini, P. Goal-Driven Requirements Analysis for Hypermedia-intensive Web Applications. Requirements Engineering Journal, Special Issue RE03, Springer 2003.
2. Grady Booch. Language Once Was Key – Now It's Design. Windows Server System Magazine. February 2003 Issue.
3. Garzotto, F., Perrone, V. On the Acceptability of Conceptual Design Models for Web Applications. In Pro. of ER'03 Workshops, (IWCMQ'03), October 2003 , Chicago, USA.
4. Mylopoulos, J., Information Modeling in the Time of the Revolution. Information Systems, Vol. 23, 1998
5. Engels, G., Groenewegen, L. Object-oriented modeling: a roadmap. In A. Finkelstein, editor, "The Future of Software Engineering", Special Volume published in conjunction with ICSE 2000, 2000.
6. Garzotto F., Paolini P. HDM- A Model-Based Approach to Hypertext Application Design. In ACM Transactions on Information Systems, Vol. 11, No1 January 1993, p1-26.
7. Isakowitz T, Stohr EA., Balasubramanian P. (1995) RMM: A design Methodology for Structured Hypermedia Design. In Communications of the ACM Vol.38 No8 August pp 34-44.
8. Schwabe, D., Rossi, G. An Object Oriented Approach to Web-Based Application Design. Theory and Practice of Object Systems, 4 (4), J. Wiley, 1998
9. De Troyer, O., Leune, C., WSDM: a User-Centered Design Method for Web Sites, in Proceedings 7th International Wolrd Wide Web Conference, Brisbane, 1997.
10. Ceri, S., Fraternali, P., Bongio, A. et al., Designing Data-intensive Web Applications, Morgan Kaufmann, 2002.
11. Hennicker, R., Koch, N. A UML-based Methodology for Hypermedia Design. In volume 1939 of LN in Computer Science, York, England, October 2000. Springer Verlag.
12. Conallen, J. Building Web Applications with UML (s.e.), Addison-Wesley, 2003.
13. Gómez, J., Cachero, C., Pastor, O., Conceptual Modeling of Device-Independent Web Applications, IEEE Multimedia, April-June 2001 (Vol. 8, No. 2).
14. Barry and Lang: A Survey of Multimedia and Web Development Techniques and Methodology Usage. IEEE Multimedia, April-June, 2001
15. C. Britton et al.: A Survey of Current Practice in the Development of Multimedia Systems. Information and Software Technology, vol. 39, no. 10, 1997, pp. 695-705.
16. B. Fitzgerald: An Investigation of the Use of Systems Development Methodologies in Practice. Fourth European Conf. Information Systems, Lisbon, Portugal, 1996
17. OMG, Object Management Group: UML 2001: a Standardization Odyssey, October 1999.
18. OMG, Object Management Group: Unified Modeling Language (UML), version 1.5 (formal/04-04-04)
19. SDTimes Magazine. UML Adoption Making Strong Progress. August 15, 2004
20. Conallen, J., Modeling Web Application Architectures with UML. Communications of the ACM, 1999
21. Kaindl, H., et al. Requirements Engineering and Technology Transfer: Obstacles, Incentives and Improvement Agenda. Requirement Engineering journal 7(3): 113-123 (2002)
22. Jackson, M., The World and the Machine. Keynote Address at ICSE-17; in Proceedings of ICSE-17; ACM Press, 1995.
23. Yu, E., Modeling Organizations for Information Systems Requirements Engineering. Proc. of the 1st Int. Symposium on Requirements Engineering, RE'93, San Jose, USA, 1993.
24. Bolchini, D., Paolini, P., Dialogue-based Design for Multichannel Interactions. In Proc. of IWWOST04 workshop held in conjunction with ICWE'04, München, Germany.
25. Baresi, L., Garzotto, F., Paolini, P. From Web Sites to Web Applications: New Issues for Conceptual Modelling. In Proc. WWW Conceptual Modeling Conf, October, 2000.

26. 24. Baresi L., Garzotto, F., Paolini, P., Perrone, V. Hypermedia and Operation Design. Deliverable D7, European IST project UWA, www.uwa-project.org
27. Perrone, V., Bolchini, D., Designing Communication Intensive Web Applications: Experience and Lessons from a Real Case. In proc. of WER 2004, 9-10 Dec. 2004 - Tandil, Argentina. To appear in a special issue on Req. Engineering of the Journal of Computer Science & Technology, autumn 2005.
28. Ghezzi, C., Jazayeri, M., Mandrioli, D.: Fundamentals of Software Engineering. Prentice-Hall, 1991.
29. Gamma, E, Helm, R., Johnson, R. and Vlissides, J. Design Patterns: Elements of Reusable Software Architec-ture. Addison-Wesley, 1995.
30. Balconi, A., Mainetti, L., Paolini, P. Perrone, V.: GENESIS-D: Formal specification of the conceptual and logical models. Politecnico of Milan, deliverable D2.2, project Genesis-D (October 2004). Available on https://www.elet.polimi.it/upload/perrone/D22ModelloConcettuale.pdf in Italian.
31. Eclipse consortium. Eclipse – Home page. www.eclipse.org/.

Towards a User-Centered Design of Web Applications based on a Task Model

Marco Winckler[1], Jean Vanderdonckt[2]
[1]*LIIHS-IRIT Toulouse,* [2]*Université catholique de Louvain*
winckler@irit.fr, vanderdonckt@isys.ucl.ac.be

Abstract

Since more than a decade, several methods for engineering and developing web applications have been introduced and extensively used. Since these methods often focus on data and related processes, their approach to conceptual modeling of web applications is centered on the notions of data, objects, functions, processes, and services. In this paper, we show that these methods could be expanded by modeling the user interface of such web applications by adopting a user-centered approach based on a task model. A task model represents the user's viewpoint on how to manipulate these data and trigger these functions so as to reach the goal associated to the task. Depending on the user, several different task models could be elaborated and each task model may lead in turn to different user interfaces for the same data and processes, as opposed to a single user interface as produced by traditional development methods. For this purpose, a case study is presented that demonstrates how a task can be modeled so as to represent the user's viewpoint in the user interface and to refine the dialog of the application.

1. Introduction

The development of Web applications is known to be as a complex activity due to many factors that need to be considered simultaneously: the evolving nature of applications, the multidisciplinary nature of development team, the competitive points of views for the application, the complexity and incompleteness of user requirements, the tight schedules for delivering the Web application [1, 6, 13]. In particular, many currently existing Web applications must follow predefined business logics and complex transactional operations and services requiring integration with distributed databases and legacy systems. Notwithstanding, the early Web was born as a hypertext/hypermedia system and it still preserves many of its hypermedia influence. As a result of this hybrid heritage, the development of many Web applications may tend to prefer considering aspects typically addressed in document management systems (e.g., information architecture), software engineering (e.g., functional architecture) [9].

To cope with this complexity, the Web Engineering community has investigated and defined models and notations intended to support the design activity of web applications that surpass the capabilities and the aims and goals of merely those found in document processing and software engineering. Several models for the development of Web applications have been proposed and extensively used in the recent years such as the OO-H method [8], the Object-Oriented Hypermedia Design Method (OOHDM) [15] and WebML [3]. Such models sometimes consist of an adaptation from previous work on Hypermedia Systems, Object Oriented methodology and Formal Methods. Those models which have been influenced by Hypertext Systems and Object Oriented methodology, propose solutions based on the concept authoring-in-the-large; which means they provide abstract models describing the overall classes of information elements and navigation structures without much concern for implementation details [7].

Most development methods existing for web applications base their conceptual modeling on the objects (or data) and their related methods, functions, or services. The consequence of this hypothesis is that the types of task that can be derived from these models frequently adopt the traditional CRUD (Create, Read, Update, Delete) pattern: tasks are limited to basic operations on objects and their relationships. When some methods go beyond this simple pattern, the dialog of the resulting user interface is assuming a transactional scheme: all data that are manipulated by the task are supposed to be entered and all methods that are required to accomplish the task should be triggered by the user. But no order is assumed, thus resulting in a dialogue where no contextual consideration is supported. For instance, such methods cannot model fine-grained dialogues such as: if the user selects this radio button, activate dynamically this push button associated to that service, enter information with this order that can dynamically change according to the user's preference, dynamically change a form according to user's reply, display this window according to the previously done operations. In other words, the dialog is often restricted to navigation between pages and screens, not to fine dialog.

Work on "classical" interactive systems has shown the central role played by task analysis for designing usable and useful systems [2, 11]. Task models allow the description of high-level user requirements in terms activities that must be performed by the user and/or by the system in order to reach some goal. The modeling

produced with task models leads to many different implementations. One advantage of this is that we can compare different design options prior to the implementation, thus saving time.

Most of the user's tasks over the Web concern the navigation [6]. By navigation we understand here all activity allowing users to move from a Web page to another, which covers supplying information through forms (identified as part of electronic procedure using a database), following a navigation paths or freely exploring the information space.

This paper argues that the use of task modeling can be employed synergistically with navigation modeling to improve the usability of Web applications designs. The Section 2 describes a general method centered on user tasks. Section 3 presents a case study for a Digital Library which demonstrates our approach. Section 3.1 presents the user roles for the application. Section 3.2 presents the modeling of user users' tasks by the means of the ConcurTaskTree notation (CTT) [14]. Section 3.3 introduces the StateWebCharts notation (SWC) [18] and presents the corresponding navigation models for the digital library. Section 3.4 shows how to drive from navigation models with SWC to the prototyping of the digital library. Section 4 presents a discussion and related work. Lately, Section 5 presents the conclusions.

2. A Method for Web User-Centered Design

We assume that the phase of requirements engineering has been already started and designers have identified the users' profile, their informational needs as well as the underlying data model. The method presented here does not impose any particular notation. We used to describe the user profile and informational requirements by textual scenarios. The underlying data model is described by the means of a UML class diagram. The approach for modeling is made up by following these steps:

1) Identify the different roles performed by the user;
2) Create a task model for each role;
3) For each task model create an individual navigation model;
4) Create relationships in the navigation model for any further informational requirements;

The step one 1 is performed in the very early phases of development when the target audience for the Web application is set up. The tasks of each user role are specified by the means of a task model at the step 2. In the step 3 task models guide the process of navigation modeling; in addition, designers can include relationships to describe system behaviors as reaction to user interaction, which is not described by task models. The step 4 is made up by detailing the navigation in order to include relationships based on informational requirements (e.g. allow users to return to the main page of the Web site whenever the page they are navigating). Additional transitions and states can be included into the model to represent single pages, external relationships, index, guided tours or any other navigational requirements.

3. Case Study: Informal Description

Our case study is the digital library of theses for the French Association on Computer-Human Interaction (AFIHM[1]). The main aim with this Web application is allowing users to navigate, search and update a digital library of theses on the HCI field. The general idea behind this Digital Library (DL) is to allow users to feed the database with little effort and control by the AFIHM. Since some users could create unexpected records (e.g. supplying incomplete/inappropriate information, mistakenly changing a record) the Web application should support a kind of review process. This review process is made up by a system administrator who decides to give or not his authorization to publish a thesis in the catalogue after have been notified by mail each time a user submit his/her thesis. The informational and functional requirements for the application can be summarized as follows:

a) To provide searching and browsing facilities;
b) All visitors can create for free their own account
c) Only users having an account can submit and download thesis from the DL;
d) To ease updating the catalogue's contents by users;
e) To support a fast review process of thesis;
f) Allow users to navigate from the Digital Library to the AFIHM's Web site;
g) Notify users about the status of their submission (i.e. accepted, refused, etc.).

Due to space restriction, only a portion of the task models and navigation models produced for the case study are presented below.

3.1 User Roles and Tasks

Table 1 presents the roles which have been identified for the application ("everyone" and "system administrator") and their corresponding (allowed) tasks. The role "everyone" corresponds to any Web site visitor, which covers the profiles "Not logged in" and "registered user". The browsing and searching facilities of the database are available to everyone but a user only can submit or download a thesis in the electronic format if s/he is logged into the system. The user role "system administrator" refers to someone who is responsible for supervising the submissions.

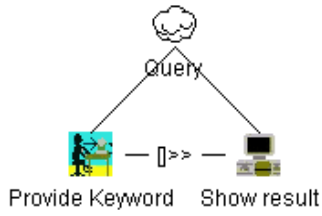
[1] AFIHM is the French acronyms of "*Association Francophone d'Interaction Homme-Machine*". More information available at: http://www.afihm.org/

Table 1. Target audience for the AFIHM's theses Web site.

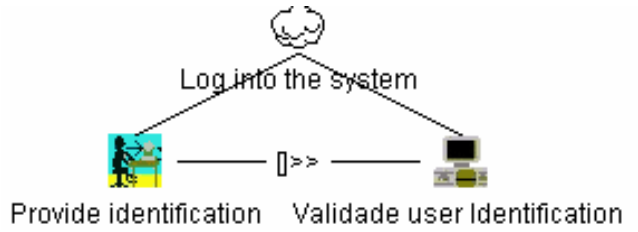
Role	User Profile	Pre-conditions	(allowed) Tasks
Everyone	Not logged in	none	Query (Search Digital Library) Create an account Log into the system
	Registered user	Logged in	Query (Search Digital Library) Download thesis Update account information Submit a thesis to the catalogue
System administrator	Have access rights Full control over the catalogue	Logged in	Review submissions

3.2 Modeling User Tasks for the Digital Library with the CTT Notation

We start by modeling individual the tasks without in-between dependencies such as "Query" and "Log into the system". Fig.1 presents these tasks by using the CTT notation [14]. In CTT, tasks are organized in a hierarchy; for example in Fig. 1.a, the task "Query" is accomplished when its subtasks "Provide Keyword" and "Show result" have been completed. The relationships between tasks are based on LOTOS operators such as enabling (operator >>), enabling with information passing (operator []>>), task interruption (operation [>), etc. CTT notation allows modeling 4 types of tasks: abstract task, user tasks, application task and interactive task. The Abstract tasks in CTT are tasks which require compound tasks such as "Query" (Fig.1.a) and "Log into the system" (Fig.1.b). User tasks are entirely performed by the user without interacting with the system (not used in this case study example). Interactive tasks are performed by the user with the system such as tasks "Provide Keyword" (Fig.1.a) and "Provide identification" (Fig.1.b). Application tasks describe actions performed by the system without user intervention, for instance "Show result" (Fig.1.a) and "Validate User Identification" (Fig.1.b). In the sequence, we designer can create more complex relationships between tasks. For example, Fig. 2 shows a complete task modeling for download a thesis form the Digital Library.



a) CTT model for the task "Query"



b) CTT model for the task "Log into the system"

Fig. 1. CTT models of individual tasks without in-between dependencies.

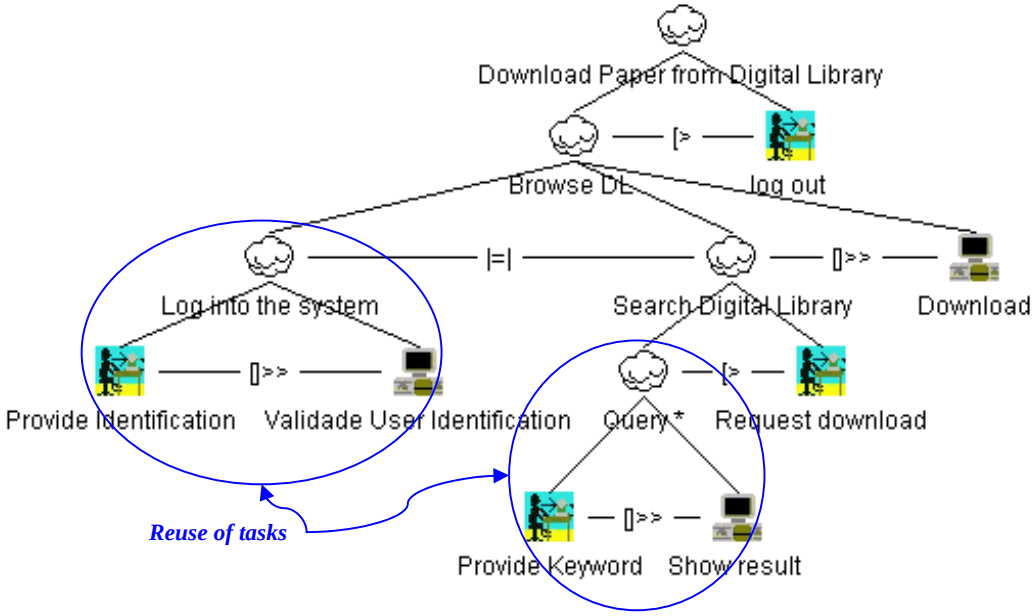


Fig. 2. CTT modeling for a download a thesis from the Digital Library.

It worth noting in Fig. 2 the reuse of tasks “Log into the system” and “Query”. The relationship (|=|) means that these tasks can be performed in any order. The system only performs the task “Send the file” after s/he has searched the digital library and get logged in the system can. In this modeling, the task “Query” is iterative (represented by the symbol *). To allow the interruption of this iteration the user can perform the task “Request download”. The task “Log out” was added to allow the users to exit the application at any time. The operator “[>” indicates the interruption of the task.

By exploiting to the task model above it is possible to perform several scenarios (see Table 2). The scenarios presented in Table 2 are used to evaluate all alternative sequences for the task “download a thesis from the digital library”. The scenarios below do not impose any particular implementation that means user tasks can be better understood without to have to planning how to support them by the system. This kind of analysis is made possible because user tasks are considered from the point of view of the users need for the application and not how to represent the user activity with a particular system.

Table 2. Some possible scenarios for the task “download a thesis from the digital library”.

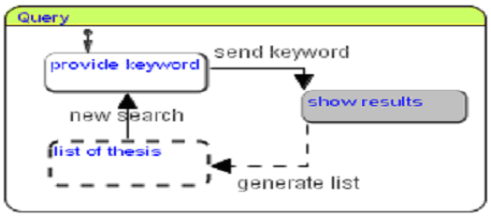
	Scenario 1	Scenario 2	Scenario 3
Sequence of task execution	Provide Identification Validate User Identification Provide Keyword Show result Request download Download	Provide Keyword Show result Provide Identification Validate User Identification Request download Download	Provide Keyword Show result Provide Keyword Show result log out

3.3 Modeling the Navigation for the Digital Library with the SWC Notation

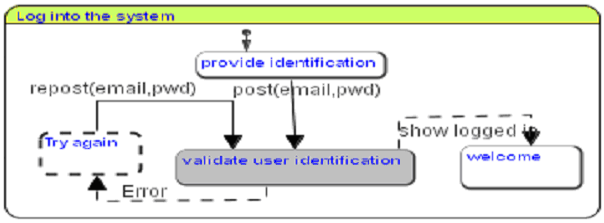
As described in previous section, some tasks require users to provide a keyword for querying a database or to provide their identification for getting access to private documents. In these cases, navigation models must represent what happens if the user identification fails or if the database records do not match to the keyword. These issues are better represented by navigation models than task models.

For navigation modeling we have proposed the StateWeb-Charts notation (SWC) [18]. SWC is a formalism based on StateCharts [10], which has been extensively used to model complex/reactive systems. StateCharts can be defined as a set of the states, transitions, events, conditions and variables and their inter-relations. In SWC, states are abstractions of containers for objects (graphic or executable objects). For Web applications such containers are usually (but not only) HTML pages. States in SWC are represented according to their function in the modeling. States can be static, dynamic, transient or external. Static states represent static content while dynamic states represent pages generated dynamically by the system. Transient states describe a non-deterministic behavior in the state machine; they are needed when a single transition cannot determine the next state for the state machine. External states represent external modules for the application of external Web sites. In a similar way, a SWC transition explicitly represents the agent activating it. Transitions whose event is triggered by a user are graphically drawn as continuous arrows while transitions triggered by system or completion events are drawn as dashed arrows. Each individual Web page is considered a container for objects and each container is associated to a state. Links and interactive objects causing transition are represented by events. The operational semantic for a SWC is: current states and their content are visible to the users while non-current states are hidden. Users can only navigate outgoing relationships (represented by the means of transitions) from current states. When a user selects a transition the system leaves the source state which becomes inactive letting the target state to be the next active state in the configuration. SWC also provides the pseudo-states as those found in StateCharts (i.e. shallow history, deep history, end state and initial state). More details about SWC can be found in [18].

In order to exemplify the elements of the SWC notation, Fig. 3 presents the SWC modeling for doing a query over the digital library and logging into the system which correspond to the tasks “Query” and “Log into the system” presented by Fig.1.a and Fig.1.b, respectively.



a) Navigation model for the task “Query”



b) Navigation model for the task “Log into the system”

Fig. 3. SWC modeling to log into the system (a) and query (b).

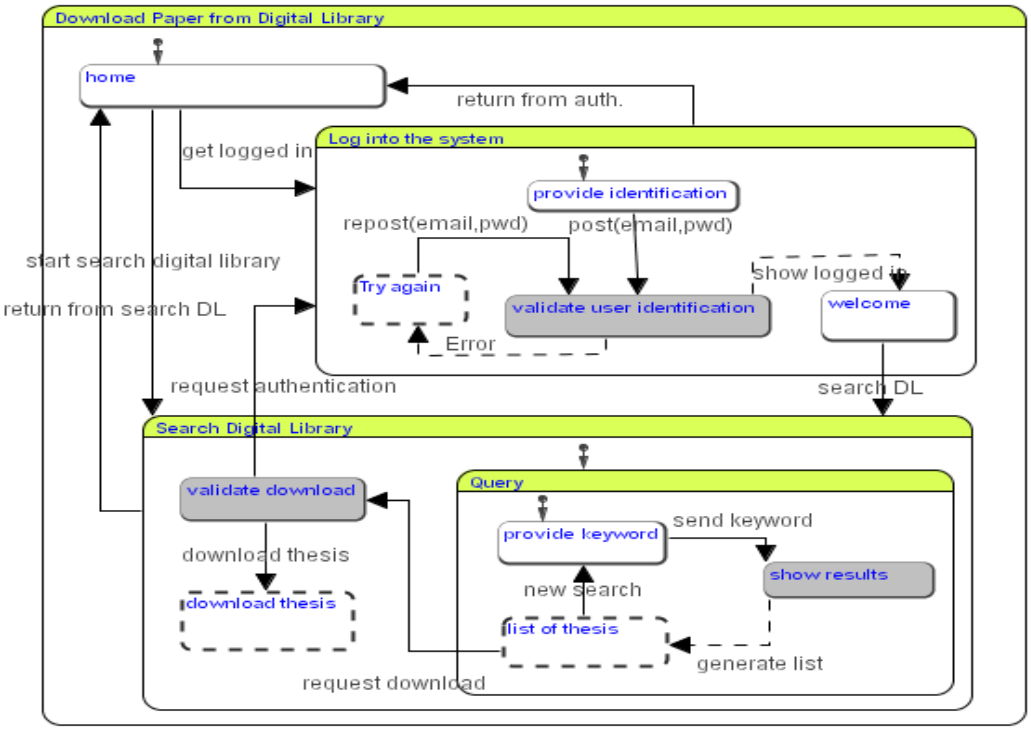


Fig. 4. SWC modeling to log into the system (a) and query (b).

In Fig. 3.b, the states “provide identification” and “validate user identification” correspond to the subtasks in Fig.1.b. The state “validate user identification” is a transient state which does have a visual representation to the user. This state is associate to dynamic state “Try again” which is dynamically generated and presented to the user if the login fails. Otherwise, the transient state present the welcome page represented by the static state “Welcome”. We can observe that these states correspond to the subtasks in Fig1.b but they also include new states (i.e. “Try again” and “Welcome”) and transitions (i.e. “Error”, “Show logged in” and “Repost(mail,pwd)” which are required to describe the behavior of the application in response to user tasks.

The Fig. 4 shows the complete navigation model for download a thesis from the Digital Library including the support for the tasks “Log into the system” and “Query” (the digital library). Similar to the previous examples, this SWC modeling includes transitions that complete the description of the system’s behavior. In addition, it features some transitions supporting content-based navigation such as “return from auth.”, “get logged in“, “return from search DL”, “start search digital library” and “Search DL”. Notice that these transitions link static states such as “home” and “welcome” which correspond to static documents. When linking up states by the means of transitions we can create all the navigation required by the users whether it concerns content-based navigation or navigation required to follows a specific procedure.

3.4 Prototyping the Web Application

The edition, simulation and prototyping of SWC models are supported by the tool SWCEditor [17] (see Fig 5). After we have verified that the navigation built with SWC holds in our requirements we can start by creating the Web pages that correspond to the SWC model. The Web pages were built using a visual environment independent from SWCEditor since, at the present, it does not integrate a Web page editor. Once we have prototyped each individual Web page for the application, we returned to SWCEditor and we associated each state to a Web page. Each state visible at the user presentation is associated to a Web page which includes the content and the graphical presentation for the objects. The SWCEditor supports the simultaneous simulation of SWC models and the execution of the corresponding Web pages. Fig. 6 provides a view at glance of this process. The navigation modeling for the digital library of AFIHM is presented at left highlighting the current state in the simulation (i.e. the state “home”). At right, Fig 6 presents the corresponding implementation of the home page. We also can observe at left of Fig 6 a dialog window showing a list of transitions going out from the state “home”. These transitions are translated to links at the home page. The arrow links indicates this translation.

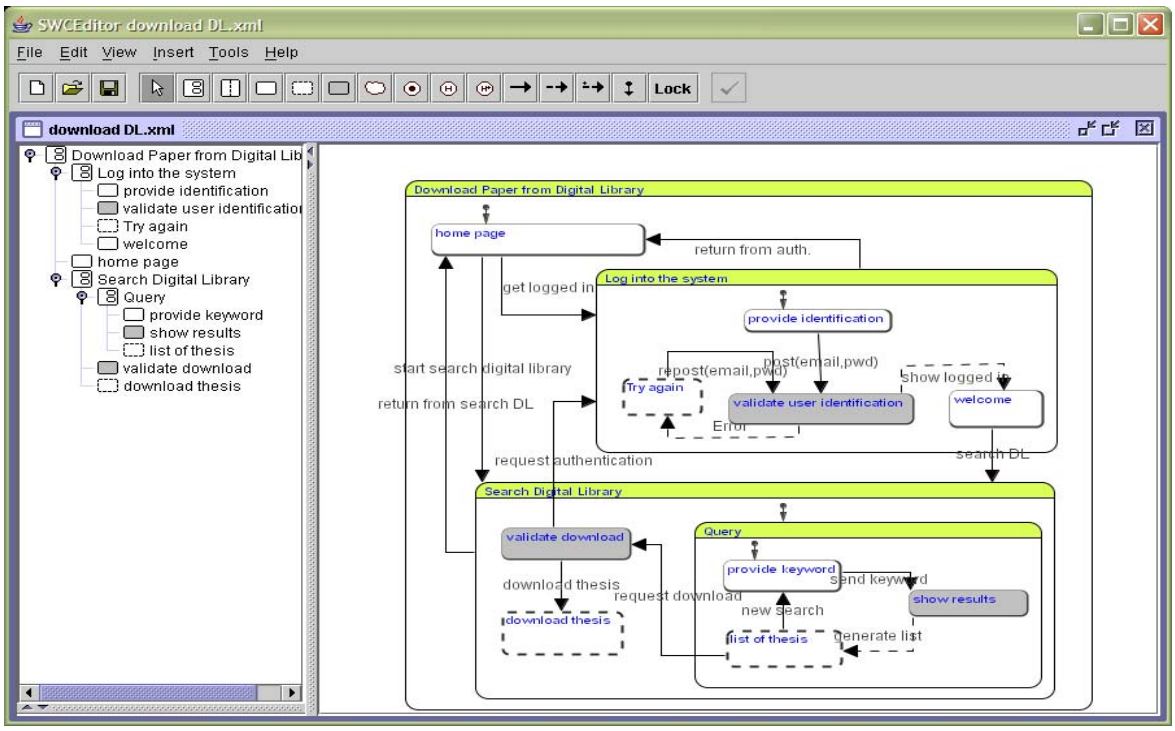


Fig. 5. SWCEditor: edition of the model “Download paper from DL Library”.

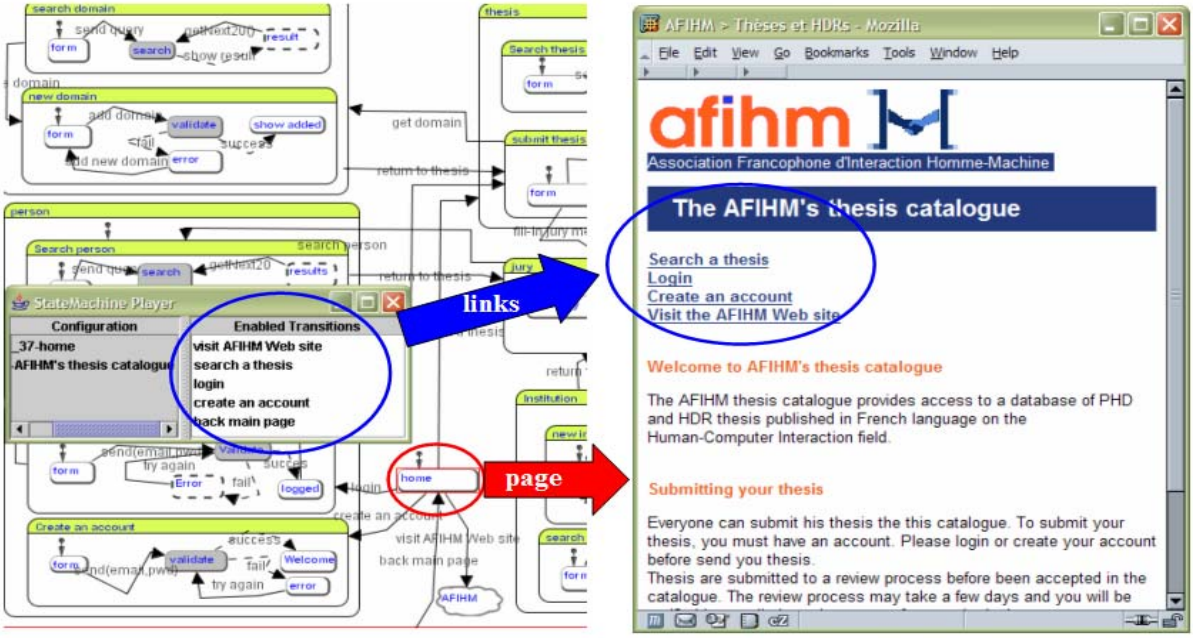


Fig. 6. Prototyping the Digital Library for the theses of the AFIHM.

4 Discussion and Related Work

When designing Web applications, we have to pay attention to the users' tasks and to the users' mental model about the information space in order to help users to navigate efficiently the application. The efficiency of Data-driven approaches is limited to the navigation one can extend from an underlying database. The main problem of such as an approach is that the underlying database does not necessarily (and quite often doesn't) represent the user mental model for the Web application. Even though task modeling is widely considered as helpful activity which let design to analyze the user activity without influence of technological constraints, the actual use of task models for the design of Web applications is underestimated mainly because current approaches for the design do not provide any guidance on how to integrate task models into the design process.

We assume that tasks models are not suitable for representing part of the system because the way users have access to information is part of the system specification not part of the user task [19]. Keep task models independent from system models allows the analysis of user needs for tasks and the transformation of such as models according to the modality and any other implementation constraints. For example, a task model should not inform how many pages a user must visit to accomplish a task because the number of the pages is on the system domain which can adapt the number of pages in the presentation dynamically according to parameters such as the device employed, the preferred modality for user interaction (graphic, sound/voice, etc), and so on. Thus both task models and system models (in the case of the Web navigation models) must be employed synergistically to produce appropriated User-Centered Designs.

As discussed in the first section, most development methods existing for web applications base their conceptual modeling on the objects (or data) and their related methods, functions, or services, and they derive tasks from the traditional CRUD (Create, Read, Update, Delete) pattern: tasks are limited to basic operations on objects and their relationships. When some methods go beyond this simple pattern, the dialog of the resulting user interface is assuming a transactional scheme: all data that are manipulated by the task are supposed to be entered and all methods that are required to accomplish the task should be triggered by the user. These development methods focus on the designer's point of view about the content and the navigation of the web application. Moreover, when the user's perspective is taken into account it is often introduced very informally. When dealing with large web application this informal process reaches its limits and often leads to usability failures.

Inappropriate navigation design of applications as one of the main sources of usability problems related to the navigation [4]. The hypertext interconnections in Web applications can be extremely complex and designers could benefit from tools and guidelines to support and assist them. Tauscher and Greenberg [16] describe some patterns of navigation but their results don't explain which tasks are engaged while these patterns are used. These studies try to describe user tasks at a high and generic (activity) level but don't provide any information about how task modeling could be performed or how a task model can be exploited within the development process of a web application.

Only a few works have been addressed the problem of model user tasks for the Web Design. The WSDM method [5] tackles many usability concerns and user requirements which are quite often neglected by other methods. However WSDM don't provide a way to model and analyze user tasks without having to take into account the system model. SOHDM approach [12] relies upon scenarios to guide all the design activities concerning the development of Web applications but it does not take into account non-functional aspects that are relevant for a user-centered design process. As mentioned before, both content-based and task-based navigation should be supported by navigation modeling methods.

5 Conclusions

This work has presented a Task-Centered Approach for navigation modeling. Our aim is to demonstrate how to describe user requirements by the means of task models and scenarios and how to transform them into navigational paths. For this purposes we have employed the CTT notation [16] to represent user tasks because it enable us to explicitly represent the tasks that are performed by the user, by the application and those which are interactive (i.e. require both system and user intervention). For navigation modeling we have employed the SWC notation [18] because it provides non-ambiguous descriptions for the navigation and it explicitly where user and system act changing the state of the application. Both notation presented are supported by tools which facilitate the edition and simulation of models.

The precise modeling of user tasks provides a deeper understanding about the user needs for the application. The mapping of task models to navigation models allows designers to explore many possible solutions for the implementations.

The role of navigation models is to create appropriate views of information space (by grouping entities from an underlying data model or a document database) and provide navigable relationships in-between according to the users' needs. Since task models are high-level description of the user activity they are not suitable to

represent all aspect concerning the navigation over Web applications. For this reason the mapping between task models and navigation models is required to complete the design.

The use of formal description techniques, such as SWC, provides a clear and non ambiguous description of the navigation supporting user tasks by the system. In addition, the appropriate tool support can alleviate the effort of modeling and support rapid prototyping, as shown in section 3.4. Even thought we wish do not discuss in this paper the verification of the SWC model, we can just mention that some tools exist to support the simulation and verification of properties of the model that can be assimilated as usability problems (e.g. deep navigation paths, dead links, and so on).

Acknowledgments

This work is partially supported by the project SPIDER Web (Capes/Cofecub n. 399/02).

References

[1] Balasubramanian, V., Bashian, A. Document management and Web technologies: Alice marries the Mad Hatter, Communications of the ACM 41(7), pp. 107-115 (1998)

[2] Benyon, D. Task Analysis and System Design: the Discipline of Data. Interacting with Computers, 4(1), pp. 102-123 (1992).

[3] Ceri, S., Fraternali, P., Bongio, A. Web Modeling Language (WebML): a Modeling Language for Designing Web Sites. WWW9 Conference (2000)

[4] Cockburn, A., Jones, S. "Which way now? Analysing and easing inadequacies in WWW navigation," International Journal of Human Computer Studies pp. 105-129, 1996.

[5] De Troyer, O., Leune, C. J. WSDM: A User Centered Design Method for Web Sites. 7th International World Wide Web Conference (1998).

[6] Fleming, J. Web Navigation: Designing the User Experience. O'Reilly & Associates Press. (1998)

[7] Garzotto, F., Paolini, P., Schwabe, D. HDM—a model-based approach to hypertext application design. ACM Transactions on Information Systems (TOIS) vol. 11, no. 1, pp. 1-26, (1993)

[8] Gómez, C.C. J., Pastor, O. Extending an Object-Oriented Conceptual Modelling Approach to Web Application Design. CAiSE'2000 pp. 79-93, (2000)

[9] Gu, A., Henderson-Sellers, B., Lowe, D. Web Modelling Languages: The Gap Between Requirements and Current Exemplars. 8th Australian World Wide Web Conference (AusWeb'2002) 2002.

[10] Harel, D. StateCharts: A Visual Formalism for Complex Systems. Science of Computer Programming vol. 8, no. 3, pp. 231-274 (1987)

[11] Johnson, P. Human-Computer Interaction: Psychology, Task Analysis and Software Engineering, Mc-Graw Hill, Maidenhead, UK, 1992

[12] Lee, H., Lee, C., Yoo, C. A scenario-based object-oriented hypermedia design methodology. Elsevier Information & Management no. 36, pp. 121-138 (1999)

[13] Murugesan, S., Deshpande, Y. Web Engineering: Managing Diversity and Complexity of Web Application Development. LNCS 2016. Springer-Verlag, Berlin, p. 355 (2001)

[14] Paterno, F., Mancinii, C., Meniconi, S. ConcurTaskTrees: a Diagrammatic Notations for Specifying Task Models, In Proceedings of IFIF T13 Conference INTERACT 97, Sydney, Chapman&Hall. (1997) pp. 362-69

[15] Schwabe, D., Rossi, G. Building Hypermedia Applications as Navigational Views of information Models. 28th Hawaii International Conference on System Sciences pp. 231-240 (1995)

[16] Tauscher, L., Greenberg, S. How people revisit web pages: empirical findings and implications for the design of history systems. Int. J. Human–Computer Studies vol. 47, pp. 97-137 (1997)

[17] Winckler, M., Barboni, E., Farenc, C., Palanque, P. SWCEditor: a Model-Based Tool for Interactive Modelling of Web Navigation. ACM Computer-Aided Design of User Interfaces (CADUI'2004), Funchal, PT. (2004)

[18] Winckler, M., Palanque, P. StateWebCharts: a Formal Description Technique Dedicated to Navigation Modelling of Web Applications. International Workshop on Design, Specification and Verification of Interactive Systems (DSVIS'2003), Funchal, PT. (2003).

[19] Palanque, P.; Bastide, R.; Winckler, M. Automatic Generation of Interactive Systems: Why A Task Model is not Enough. 10th International Conference on Human-Computer Interaction – HCI International'2003, Héraklion, Greece, June 2003.

Defining a UML Profile for Web–based Educational Applications

Andreas Papasalouros
National Technical University of Athens
Software Engineering Laboratory
9 Heroon Polytechneiou, 15780, Zografou, Greece
andpapas@softlab.ntua.gr

Symeon Retalis
University of Piraeus
80 Karaoli & Dimitriou,
18534, Piraeus, Greece
retal@unipi.gr

Abstract

This paper presents a UML profile for web–based educational applications. The definition of the profile is provided by applying a certain formalism which is based on the metamodeling architecture of the UML language. An example is given throughout the paper for the illustration of an instantiated model based on the profile.

1. Introduction

The World Wide Web is becoming the basis of ubiquitous, learner–centered learning. Time and space independence and the ability for multiple representations of learning material due to enhanced multimedia presentation capabilities are among the characteristics of the Web that enforce its potential as a medium for education and training.

Web–based Educational Applications (WEA) are important components for web–based learning. By the term 'educational application' we mean properly structured and presented hypermedia material composed of educational resources and delivered through the web. The presentation of these resources is usually performed by specialized runtime environments, web applications or integrated systems for the support of learning through the web called Learning Management Systems. An trivial case of an educational application is an electronic book containing resources as web pages with educational content. The delivery of resource can be static or dynamic, that is, adapted to the user interaction with the application, though the latter case is not considered in this paper.

The development of WEA is a complex task. It involves people with different background such as software developers, web application experts, content developers, domain experts, instructional designers, etc. Furthermore, WEA are complex dynamic web–based applications with presentational, behavioral, architectural aspects. In order to effectively capture and specify the various aspects of a WEA, to document the decisions concerning various aspects of such applications, from the implementation of pedagogic and instructional design to subtle technical decisions of such applications and to facilitate the communication between the members of usually heterogeneous development teams there is a need for a design model. Experience from traditional software engineering has shown that the adoption use of such a model in the context of a systematic process for the development is beneficial for the quality of the product, namely educational applications and the efficient management of the resources, time and effort. Thus, a number of modeling approaches for web–based applications have been proposed, e.g. OOHDM [11], RMM [4], WebML [2]. Furthermore, specific models for educational applications exist. EML [5], which has evolved to the Learning Design specification [3] proposes a modeling formalism for educational environments. EML does not aim to support the design but rather to provide a conceptual framework for the description of educational settings as a set of "units of study". LMML [12] is another modeling language for educational content, which can be used for design in this domain.

This paper presents a formal definition of a modeling language for design models for WEA. This language is defined as a sub–set, i.e. a profile of the Unified Modeling Language (UML), a de facto standard for the modeling of software systems. It focuses on the formalism used for the definition of this new modeling language and not in the language itself, which is described elsewhere [10]. The purpose of the paper is to provide an overview and not a full specification of this language. The structure of the paper is as follows: In the next section a discussion of UML profiling and the approach adopted for defining this profile is presented. Next, the modeling language for WEA is described together with the definition of the profile for this language. The paper ends with a discussion of the consequences of formal modeling and some extensions of this work.

2. UML metamodels and profiles

The proposed modeling language is based on the UML. It is an extension of the UML by introducing new modeling elements and specified as a UML Profile. According to the UML specification a UML profile is "a coherent set of extensions, defined for specific purposes" [9]. More generally, profiling is the customization of a general purpose technology for a particular domain [6]. This is a process involving the elicitation of relevant only elements of the generic technology and the extension of the semantics of these elements towards the particular domain of application. In this way, the generic technology becomes more efficient and easy to apply.

The UML is a visual language. The definition of the language itself is based on a 'Four–Layer Metamodel Architecture' [9]. According to this approach, the definition of the language is structured in four layers: Meta–metamodel, metamodel, model and user objects. Each layer acts as a meta–language for the definition of the next layer. From another perspective, each layer is an *instance* of the previous one.

- The meta–metamodel layer provides the abstract constructs for the definition of the metamodel layer, e.g. MetaClasses, MetaAttributes, MetaOperations, etc. The definition of this layer is based on the Meta–Object Facility (MOF) [7] which is a specification for the definition and interchange of metamodels.
- The metamodel layer provides a language for specifying actual models. Examples of metamodel elements are Classes, Attributes, Operations, etc. Metamodel elements are represented as (meta) classes. The metamodel contains a set of UML class diagrams which contain metaclasses connected with association and generalisation relationships. Furthermore, for the definition of valid models additional rules and constraints must be provided. These rules and constraints are applied as "well–formedness" rules, which are also part of the UML metamodel. These rules are expressed in a Object Constraint Language (OCL) [13], which is also part of the UML specification, and, in few occasions, in natural language.
- The model layer contains instances of the modeling elements of the metamodel which are actually models of a particular application or domain, for example a class Course belongs to the model of a learning system and it is an instance of the Class element of the metamodel layer.
- Finally, the user objects layer defines specific information for the application domain into consideration, for example the particular state of a Course class instance.

Our purpose is to define a language for describing models for the application domain of web-based educational applications. The structure of these models and the concepts in use are designated by the application domain under consideration. As mentioned above, this language is defined by means of a UML profile. The establishment of a UML profile is provided by extending the metamodel layer in the four–layered architecture. This extension is provided by using the standard extension mechanisms of UML, namely stereotypes, tagged values and constraints. The basic extension mechanism, stereotypes, extends the existing UML elements, belonging to the metamodel layer, thus widening the vocabulary of the language. However, the mere definition of the UML extension mechanisms mentioned before does not provide a complete Profile, adequate for modeling a new domain by capturing both its concepts and their relationships in a consistent and meaningful manner. The approach that we follow for defining a profile for WEA is based on the UML Profile for CORBA Specification [8]. According to the latter, a profile consists of the following:

- The extension of a subset of the UML metamodel with new stereotypes and the assignment of these stereotypes to existing UML (metamodel) elements. This constitutes a "Virtual Metamodel".
- The definition of additional of well–formedness rules, beyond these defined in the UML metamodel, for the new modeling elements. These rules impose additional constraints to the new elements so as to express the requirements of the particular domain. These rules refine and do not contradict with the rules that apply to their base metamodel elements, so as not to alter their semantics.
- The specification of the semantics of the model in natural language.

The well-formedness rules for the elements of the profile are defined as constraints expressed in the OCL and, in few occasions, in natural language. OCL is a language for the expression of constraints in UML model elements. As mentioned before, new constraints are defined for the new elements defined in the profile. The well–formedness rules expressed using OCL invariants. An invariant expression applies to a particular *context*. The context in which an invariant condition applies is denoted by the following expression which is in the start of every invariant:

```
context [Element Name] inv:
```

where `[Element Name]` is the name of the element under consideration to which the invariant applies. In order to define these rules proper OCL operations are

needed. The operations used are OCL 'built–in' operations, operations defined in specification documents such as [9] and [8] or new operations defined for the purpose of defining the profile. These operations are presented next. The OCL expressions which define these operations are referring to the UML Relationships metamodel. The `aggregatedClasses` operation returns the set of classes (classifiers in the UML metamodel more generic terminology) that are connected with the class under consideration through composite aggregation associations:

```
aggregatedClasses: Set(Classifier)
aggregatedClasses =
 self.associations->
  select(a | a.connection->
   select (ae |
    ae.participant = self
     and
    ae.aggregation =
     #composite))->
  collect (ae |
   ae.connection->
    select(ae |
     ae.participant <> self)->
      collect(p |
       p.participant)).asSet
```

The `aggregateClasses` operation returns a set of Classes that are at the opposite side of aggregation associations, i.e. they contain the particular class:

```
aggregateClasses: Set(Classifier)
aggregateClasses =
 self.associations->
  select(a | a.connection->
   select (ae |
    ae.participant = self ))->
  collect (ae |
   ae.connection->
    select(ae |
     ae.participant <> self
      and
     ae.aggregation =
      #composite))->
       collect(p |
        p.participant)).asSet
```

3. Description of the profile for WEA

In the next sections the following sub-models are defined as UML Packages: Conceptual Model, Navigation Model and User Interface Model.

3.1. The Conceptual Model

The Conceptual Model captures design decisions during Instructional Design. These decisions are described as a hierarchy of learning activity and associated resources. An example of such a model is illustrated in Figure 1.

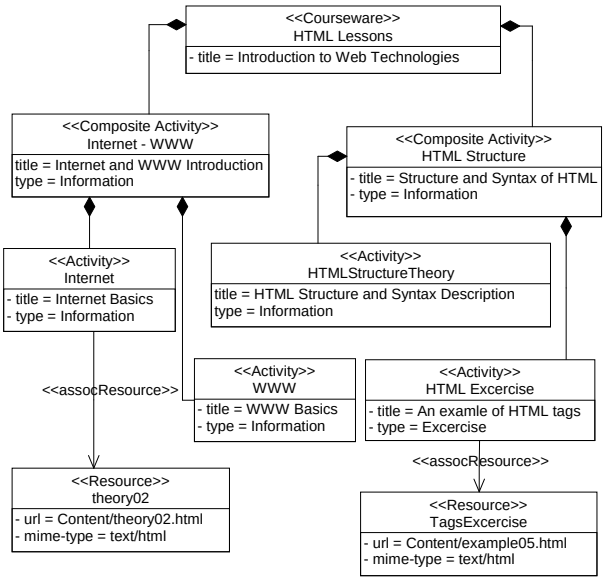


Figure 1. Example of a Conceptual Model

The virtual metamodel for the conceptual model is shown in Figure 2. In this figure the Conceptual Model is shown as a class with stereotype named `stereotype`. This (meta)class is connected with a dependency relationship with stereotype `baseElement` with a base class named `ModelManagement::Model`. This configuration denotes the fact that the element Conceptual Model 'instantiates' the element Model of the UML metamodel, i.e. it is a Model extended by stereotyping. The prefix `ModelManagement` states the fact that the metaclass Model belongs to the ModelManagement package of the UML metamodel. Figure 3 shows the elements of the Conceptual Model, namely `Activity`, `Composite Activity`, and `Resource`, which are stereotyped classes. `Abstract Activity` is a generic type of activity defined as an abstract metaclass, which means that `Abstract Activity` does not appear as a modeling element. The subclassing mechanism is used for organisation purposes. Rules and constraints applied to superclasses are inherited to the subclasses, unless overridden by new rules applied to the descendant classes. Composite activities contain other activities by connecting with them with aggregate associations. In this way, hierarchies of activities are defined in the Conceptual Model. A special type of composite activity with stereotype `Courseware`

is the root in the activity hierarchy. In addition, atomic activities are related with one or more Resources with `AssocResource` associations (See Figure 4).

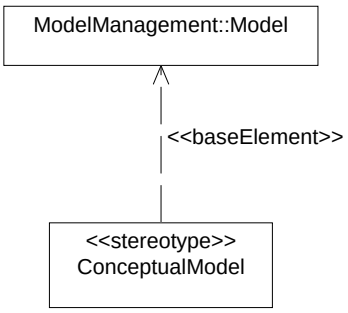


Figure 2. Conceptual Virtual Metamodel

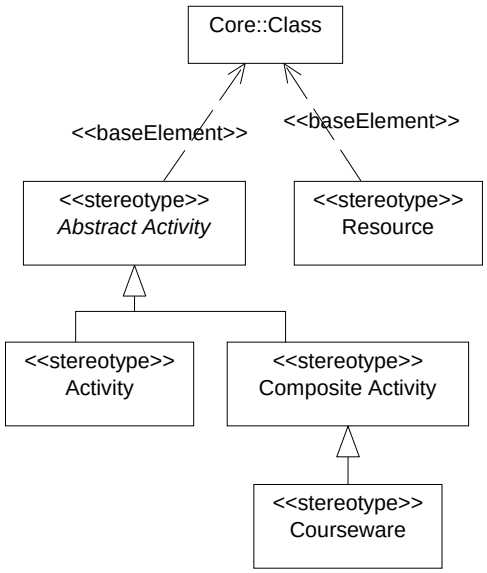


Figure 3. Conceptual Model Classes Virtual Metamodel

- The Conceptual Model contains only Activity and Resource classes

```
context ConceptualModel inv:
 self.ownedElement->forAll(e |
  e.oclIsKindOf(Class)
   implies (
e.isStereokinded("AbstractActivity")
 or
e.isStereokinded("Resource"))
```

- Abstract Activities have a `name` and a `type` attribute.

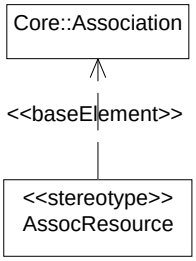


Figure 4. Conceptual Model Associations Virtual Metamodel

```
context AbstractActivity inv:
 self.features->
  select(a |
   a.oclIsKindOf(Attribute)
   and
   a.name = 'name').
    size = 1
 and
 self.features->
  select(a |
   a.oclIsKindOf(Attribute)
   and
   a.name='type').
    size = 1
```

- A Resource has a `mime-type` and a `url` attribute. The expression for this constraint is syntactically similar to the above and it is omitted.

- Composite Activities are associated (through aggregation associations) with activities, either atomic either composite.

```
context CompositeActivity inv:
 self.aggregatedClasses->forAll(c |
c.isStereokinded("AbstractActivity"))
```

- Atomic activities do not contain other classes through aggregation associations.

```
context Activity inv:
 self.aggregatedClasses.isEmpty
```

- A Courseware is not contained by any other classes through aggregation associations.

```
context Courseware inv:
 self.aggregateClasses.isEmpty
```

- `AssocResource` associations connect one `Resource` stereotyped class with a `Concept` stereotyped class.

```
context AssocResource inv:
 self.connection.size = 2
 and
 self.connection.participant->
  exists(c |
   c.isStereokinded("Activity"))
 and
  exists(c |
  c.isStereokinded("Resource"))
```

3.2. Navigation Model

The Navigation Model captures the design decisions concerning the definition of the actual web pages of the learning content, as well as the links that facilitate the navigation through these pages.

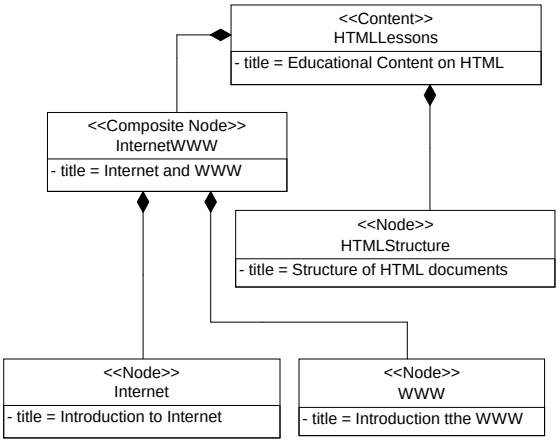


Figure 5. Example of a Navigation Structure Model

The navigation model is separated with two sub-models: The Navigation Structure Model deals with the definition of the actual web pages, the links that facilitate navigation as well as the definition of the actual content of these pages. The Navigation Behavior Model defines the dynamic, adaptive behavior of the WEA. In this version of the profile deals with static applications only so the Behavior Model is not considered. The Navigation Structure Model defines the (static) navigation structure of a WEA by means of the following elements: `Content`, which is a top-level container, `Composite Node`, which is a composite elements containing other elements, such as a chapter in an electronic book, `Nodes`, which are the actual web pages. Again, `Abstract Node` is an abstraction of a composite or atomic Node. The structure of the nodes is hierarchical, as the Conceptual Model. In addition, `Link` elements can connect nodes of the navigation model. Note that these links are associative links that define arbitrary navigation between nodes of the educational hypertext. Besides these, implicit structural links exist. These links denote the transitions to the previous or next node in the traversal of the hierarchical structure of nodes or the selection of a certain node. The presentation of these links is supported by the run–time system which delivers the educational application. Although both conceptual and navigational model diagrams are hierarchical, the corresponding graphs are not necessarily isomorphic. Nodes are realisations in the navigation space of one or more learning activities. Thus, one node can be related to one or more activities. Figure 5 illustrates an example of a Navigation Structure Model, which is an instantiation of the above virtual metamodels. The trace relationships between elements of this diagram and the Conceptual Model are not depicted, since they are obvious from the names of the elements.

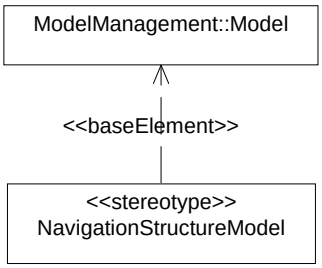


Figure 6. Conceptual Virtual Metamodel

The following constraints are applied to elements of the Navigation Structure Model:

- As mentioned above, the structure of nodes is hierarchical, as is the Conceptual Model. Thus, similar constraints apply to its elements as regards to aggregation. A `Composite Node` stereotyped class is connected through aggregation associations with `Abstract Node` classes, `Content` class has no aggregate classes, i.e. it is not contained by any other node, and, finally, atomic nodes do not contain other nodes. The OCL expressions for these rules are omitted for the sake of brevity.
- The Navigation Model contains only `Abstract Node` classes.
- A `Link` stereotyped association connects Nodes. A link represents a unidirectional link from a source to one or more destinations. It denotes the direction of

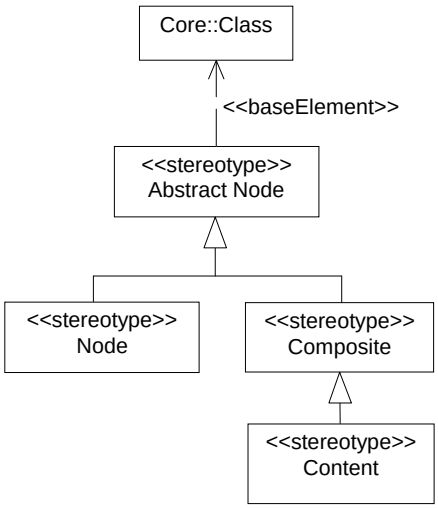


Figure 7. Navigation Model Classes Virtual Metamodel

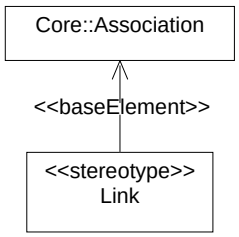


Figure 8. Navigation Model Associations Virtual Metamodel

a link in the following way: In the UML metamodel an association contains two or more association ends. These ends have a navigability attribute. Navigability is graphically depicted with an arrow at the navigable end of the association. The navigability of the association end, that is the direction of the arrow in the visual presentation of the association, denotes the direction of the link. Thus, there is exactly one end which is not navigable, which corresponds to the "source" of the link and one or more navigable ends, which correspond to the "destination" of the link.

```
context Link inv:
 self.connection.participant->
  forAll( p |
   p.isStereokinded("Node"))
 and
 self.connecions->
  select(ae |
      ae.isNavigable = false).
       size = 1
```

- `Abstract Node` elements have an attribute named `title`.
- An `Abstract Node` is related to one or more `Abstract Activity` elements of the Activity Model with a Trace dependency. This is a standard UML element, a special kind of relationship which depicts dependencies model elements between different models. This constraint is provided in natural language only.

3.3. User Interface Model

The User Interface Model deals with the presentation aspects of the elements defined in the Navigation Model i.e. their layout and style. In particular, nodes of the Navigation Model are associated with an element of the User Interface model. Multiple navigation elements can be associated with the same presentation specification, thus promoting uniformity and maintenability of the user interface. The User Interface Model elements have their counterparts in corresponding elements of web technology specifications, namely HTML and Cascading StyleSheets (CSS) elements. The basic element in this model is named `Template`. A template contains the following stereotyped UML classes with aggregation associations: `Html`, that represents HTML elements or aggregations of HTML elements and `Css` that represent Cascading Style Sheet classes. Attributes of these classes and their values are attributes and values of CSS elements. Elements defined in the User Interface model are related with Trace dependencies to particular nodes of the Navigation Model thus assigning specific presentation attributes to these nodes, as well to their children in the navigation structure hierarchy. No metamodel diagrams are shown, since they are similar to those of the Conceptual and Navigation Structure models. Figure 9 shows an example of a User Interface Model for the HTML educational application under consideration. The trace relationship between the template and the root element of the navigational model means that all nodes (pages) share the same template.

The following constraints are applied to the User Interface Model elements:

- The Presentation Model contains `Template`, `Css` and `Html` classes.
- `Template` classes contain, through composite aggregations, `Css` and `Html` elements.
- `Template` classes are connected with Abstract Nodes of the Navigation Model through `Trace` relationships.

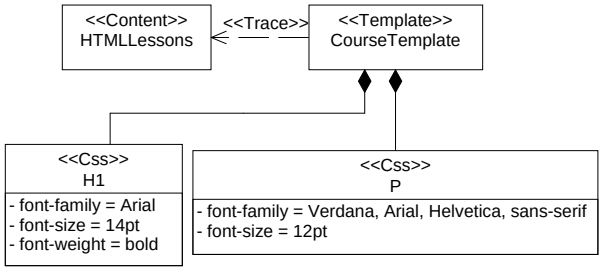


Figure 9. Example of a User Interface Model

4. Discussion

The definition of a Design Model can facilitate the process of developing software projects, regardless of the domain of the application. This facilitation is even more important in fields where the people involved in the development process come from different backgrounds so there is an increased need for a means of communicating design decisions. However, this improvement is often confuted by the lack of formalism in the definition of such models. This lack of formalism has certain negative aspects:

- Poorly defined models, which are based on the intuition of the designers rather than in predefined 'rules'.
- It is impossible to automate the authoring of models by means of specific CASE tools.
- It is impossible to automate the process of automatic code generation based on the models created (forward engineering).

Rigor definitions of modeling languages alleviates the aforementioned problems. A definition of a modeling language is usually provided by means of a metamodel, i.e. a model of the set of all valid models in the language. Such a MOF-based metamodel for web application design is proposed in [1]. Furthermore, EML and LMML presented earlier, are defined by means of a metamodel, i.e. a normal UML model that describes the modeling primitives of the language. Besides modeling primitives, EML and LMML provide an XML definition and provide appropriate schemas, as an additional formalism. Conversely, our approach is based on the extension mechanisms of UML for the creation of a profile, thus it does not define a new metamodel but rather refines the existing metamodel of the language using a specific formalism.

We are in the process of extending the definition of the profile so as to include dynamic features of WEA. Furthermore, this profile was defined based on version 1.5 of the UML and OCL. We intend to update it by using the version 2.0 of these languages.

5. Acknowledgements

This work was partially supported by the "TELL: Towards effective network supported collaborative learning" project which is partly sponsored by the European Commission under e-learning program (ref num: 2003-4721 TELL).

References

[1] L. Baresi, F. Garzotto, L. Mainetti, and P. Paolini. Metamodeling techniques meet web application design tools. In R.-D. Kutsche and H. Weber, editors, *FASE*, volume 2306 of *Lecture Notes in Computer Science*, pages 294–307. Springer, 2002.

[2] S. Ceri, P. Fraternali, and A. Bongio. Web Modeling Language (WebML): a modeling language for designing web sites. *Computer Networks*, 33(1-6):137–157, 2000.

[3] IMS. *IMS Learning Design Information Model, Version 1.0 Final Specification*, Jan. 2003.

[4] T. Isakowitz, E. A. Stohr, and P. Balasubramanian. RMM: A methodology for structured hypermedia design. *Communications of the ACM*, 38(8):34–44, Aug. 1995.

[5] R. Koper. *Modeling Units of Study from a Pedagogical Perspective: The Pedagogical Meta-Model Behind EML*, 2001.

[6] R. Malveau and T. J. Mowbray. *Software Architect Bootcamp*. Prentice Hall PTR, Upper Saddle River, NJ, 2001.

[7] OMG. *Meta Object Facility (MOF) Specification, Version 1.4*, Apr. 2002.

[8] OMG. *UML Profile for CORBA, Version 1.0*, Apr. 2002.

[9] OMG. *Unified Modeling Language Specification, Version 1.5*, Mar. 2003.

[10] S. Retalis and A. Papasalouros. Designing and automatically generating educational adaptive hypermedia applications. *Educational Technology & Society*, to appear.

[11] D. Schwabe and G. Rossi. The object-oriented hypermedia design model. *Communications of the ACM*, 38(8):45–46, 1995.

[12] C. Süß and B. Freitag. Metamodeling for web-based teachware management. In P. P. Chen, D. W. Embley, J. Kouloumdjian, S. W. Liddle, and J. F. Roddick, editors, *Advances in Conceptual Modeling: ER '99 Workshops on Evolution and Change in Data Management, Reverse Engineering in Information Systems, and the World Wide Web and Conceptual Modeling, Paris, France, November 15-18, 1999, Proceedings*, volume 1727 of *Lecture Notes in Computer Science*, pages 360–373. Springer, 1999.

[13] J. Warmer and A. Kleppe. *The Object Constraint Language: Precise Modeling with UML*. Addison Wesley, Reading, MA, 1999.

Instantiating Web Sites Quality Models: an Ontologies driven Approach

Luisa Mich
Department of Computer and Telecommunication Technology
University of Trento, Italy
{luisa.mich}@unitn.it

Mariangela Franch
Department of Computer and Management Sciences
University of Trento, Italy
{mariangela.franch}@unitn.it

Abstract

One of the most important steps in a Web site quality evaluation project is the selection of which aspects to consider. In terms of methodology, this means defining a model for the site. In some cases it is possible to use standardized models, such as "syntactic" models, but this is not possible when the evaluation must also consider aspects that have to do with the domain and the specific aims of the site or more generally when the evaluation aims to consider the "semantics" of the site. The process of identifying and adapting a quality model requires, apart from time and resources, the contribution of experts in the domain of the site. In this paper we propose to use ontologies to improve the efficiency of this "instantitation processs". To analyze the feasibility of the approach we have looked at two applications in the tourism sector. The results, while preliminary, are encouraging. Moreover, some critical and delicate points were identified as priorities for future research.

1. Introduction

Quality in Web sites is determined by several diverse factors, some of which are general and therefore are considered for all types of site and in all domains. Such features include, for example, the correct functioning of the site, its conformity with standards of language use or of accessibility as described in normatives such as the Web Content Accessibility Guidelines of the W3C (http://www.w3.org/WAI/GL/) or the U.S. Section 508 Guidelines (http://www.section508.gov/), or standards introduced in Italy through the Stanca law in 2004 (http://www.pubbliaccesso.it/biblioteca/documentazione/studio_lineeguida/), which requires these standards for government sites. Other factors or characteristics are more specific and depend on the type as well as the domain of the site. Therefore in cases where it is not possible to use "standard", "syntactic" models - general-purpose and domain-independent – it becomes necessary to develop quality models that take also these features into consideration (among models having a standard version we can look at WebQual [2] and WebQEM [11]; an extensive bibliography of models for Web site quality is available at: http://www.economia.unitn.it/etourism/risorseQualita.asp). Basically this means defining specialized models that can deal with the unique semantic aspects of a site or sites that will undergo evaluation. The process of definition and instantiation of a model takes time and resources and also the input of experts in the domain. In this paper we propose adopting an approach based on the use of ontologies to support the definition of detailed semantic models. This approach is an extension of a methodology "the Quality Model Factory" described in [9] and successfully applied in the area of tourism. It was applied here to define modular quality models that make it possible to take into account the characteristics of diverse types of tourist destinations. To do this the models were developed using a standard model called the 7Loci meta-model [10]. There are two types of "modules" specialized at two different levels of detail: the first, called the Common module, contains aspects that are common to the sites of all types of tourist destinations, while the second is comprised of Specialized modules that contain specific aspects that are found at different types of destinations. To analyze the feasibility of a methodology based on the use of ontologies to define specialized models for Web site quality evaluation, we looked at the sites of accommodations and of tourist destinations facilities. Both are within the tourism domain and are highly varied given the diversity of types of tourist destination as well as accommodations. Moreover, the decision to look at the tourist sector – a transversal sector that includes numerous actors and activities – made it possible to have a general idea of the difficulties and challenges, as well as the advantages, of using ontologies to define quality evaluation models that are specialized and modular.

Recent years have seen increased interest in the development and application of ontologies. This has

meant firstly the definition of languages and environments to set up ontologies. Examples of languages are DAML (Description Logic Markup Language), OIL (Ontology Interchange Language), RDF (Resource Description Framework) and OWL (Ontology Web Language), a semantic markup language for publishing and sharing ontologies on the World Wide Web (http://www.w3.org/TR/owl-ref/). Among the more numerous applications of ontologies are projects related to the Semantic Web, to obtain "an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation." [3]. An obvious application of ontologies is in Web services; among the most recent works and most frequently downloaded (was in the fifth position of the Top 10 Downloads from ACM's Digital Library in December 2004), we can cite [12]. Also important to note is the use of ontologies for the development of software based on reuse [5] and for the management of multimedia objects in a private and personal environment [7].

The proposal described in this paper differs from these insomuch as it is applied at a different level: it is designed to use ontologies to support the development of quality models that are specialized for Web sites. As such it is a conceptual as well as methodological activity, and is applied at a meta-level with respect to the application of the models themselves.

The paper is structured as follows: the first section gives a general description of ontologies for tourism. Following this we look at the concept of tourist destination, where we present the classification used to apply the "Quality Model Factory" methodology, which we are going to extend to include domain ontologies. The third section puts forth two possible applications for the development of specialized and modular evaluation schemes, respectively for hotel Web sites and for tourist destinations. The conclusion summarizes the preliminary results of these applications, underlining the critical points emerging from the application of the methodology and which require further study.

2. Ontologies for tourism

Tourism represents approximately 11% of worldwide GDP, according to the World Travel & Tourism Council (http://www.wttc.org/). Adding to this is the notable growth in the number of tourism-related Internet transactions in recent years (e-commerce). For example, in 2004, 40% of U.S. travelers who use the Internet claimed to make all of their travel purchases online, versus 29% in 2003 (www.tia.org). In this context, the quality of Web sites becomes a vital strategic factor for all actors involved. Because tourism is a transversal sector – or "umbrella industry" – it has contributions from other sectors, thus an analysis of the sector must have input from different fields such as transport, culture, and sport, to name a few. This fact explains the existence of numerous ontologies for tourism. An exhaustive classification can be found in [1].

Existing ontologies are both general for the tourism sector as well as specific, the latter referring to particular domains. In the first category we have the ontology developed for the Harmonise project, whose goal is to develop an ontology-mediated integration of tourist systems following different standards so that organizations can exchange information without changing their data structures (http://www.harmo-ten.info/). In addition there is the Mondeca´s tourism ontology, which includes tourism concepts from the WTO (World Tourism Organization) thesaurus. At this writing this ontology has 1000 concepts that describe accommodations and transportation and a few other secondary elements related to geography, health and immigration (http://www.mondeca.com/).

There are over ten elements on the list of domain-specific ontologies that can be useful for the tourist sector, including geographic ontologies, means of transportation ontologies, gastronomy ontologies, etc. [1].

General – or sometimes called upper – ontologies also exist and aim to gather definitions and concepts that together make up what is known as unspecialized common knowledge. One of the best known of these is WordNet – more appropriately referred to as a lexical reference system (http://www wordnet.princeton.edu/) which was extended from solely English into other languages through the EuroWordNet (http://www.globalwordnet.org/gwa/wordnet_table.htm).

Looking at the ontologies for hotels and tourist destinations (the organizations examined for the feasibility analysis of the approach proposed here) the following facts emerged.

Since the concept of hotel is part of common knowledge, the notion is present in WordNet. More specifically, for each concept – in this case *hotel* – WordNet gives information on the generalization, specialization and also on "part of" relationships. A description of the linked concepts is available at Answer.com (www.answer.com). In short, in WordNet, focusing to concepts that are directly linked to *hotel*, thus exploring only relationships represented by arcs going out from the node of hotel, we obtain (see also figure 1):

hotel is a kind of:
- building, edifice – a structure that has a roof and walls and stands more or less permanently in one place

kinds of hotel:
- hostel, hostelry, inn, lodge – a hotel providing overnight lodging for travelers
- motel, motor hotel, motor inn, motor lodge, tourist court, court – a hotel for motorists; provides direct access from rooms to parking area
- resort hotel, spa – a fashionable hotel
- Ritz – (informal) an ostentatiously elegant hotel
- ski lodge – a hotel at a ski resort

Parts of hotel:
- hotel room – a bedroom (usually with bath) in a hotel

Furthering the analysis by using the concept of *hotel room*, WordNet gives other information:

kinds of hotel room:
- adjoining room – a hotel room that shares a wall with an adjoining room but is not connected by a door
- connecting room – a hotel room that shres a wall with an adjoining room and is connected by a private door

hotel room is a kind of:
- bedroom, sleeping room, chamber, bedchamber – a room used primarily for sleeping.

hotel room is a part of:
- hotel – a building where travelers can pay for lodging and meals and other services

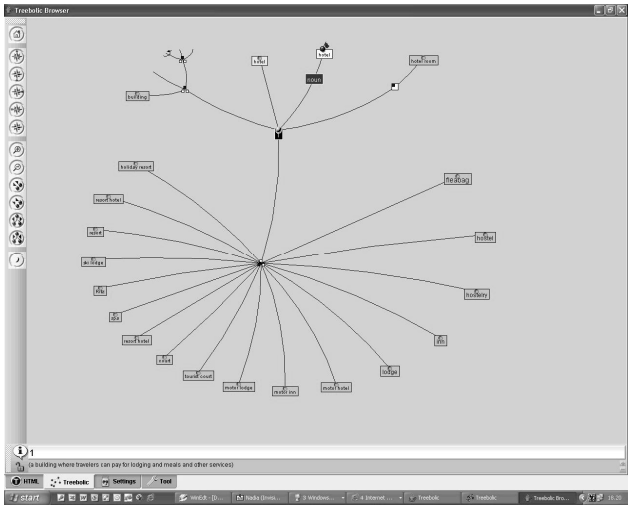


Figure 1 – The graph for *hotel* (WordNet browser Treebolic, http://treebolic.sourceforge.net/)

The quantity of information about the concept of *hotel* contained in WordNet is exhaustive, an important fact since WordNet is free and has been used in many projects. Answers.com also provides hypertext definitions (where they exist) found in other sources such as Wikipedia (http://www.wikipedia.org/), which can be useful to integrate with those contained in WordNet, and also their translation into diverse languages.

The more challenging problem is to identify an ontology for tourist destinations, although this difficulty is justified by the complex definition of destination, a concept which, albeit only recently, is assuming an increasingly important role in the tourist sector in general. Basically, the features that identify a tourist destination and distinguish it from what is simply a local offering of a product or service that can be of interest to tourists are:

- a well-defined geographic area with identifiable borders and a territorial identity;
- the presence of numerous operators with different prospectives and objectives that makes it necessary to devise a shared strategy in presenting the offering consisting of attractions and services specifically catering to tourists in the location;
- an understanding of the nature of the potential demand for the tourist products offered;
- awareness of the need to balance tourism's exploitation of resources with ecological, environmental and community stewardship.

A classification of the destinations serving the leisure tourist segment identifies eight distinct types of destination based on the goals for the vacation and the principal attractions present at the destination (table 1) [8]. The table shows key information about the defining features of a destination. Once established, these aspects can then serve as input when determining the requirements and the quality factors for the Web site of the destination.

The definition of tourist destination and the classification of diverse types of destination show how the necessary concepts belong to diverse domains. WordNet gives no treatment to the concept of tourist destination. Numerous ontologies were examined as part of this research, but none proved able to cover all the elements that characterize a destination: it is thus necessary to use different ontologies to describe the geographic area, the different attractions, sports, transport, etc. On the other hand, general ontologies (upper ontologies) contain a lot of information that is not useful because it is related to concepts that differ greatly from those used in the tourism context. In addition, general ontologies for tourism cover only some of the

necessary areas (for example, Mondeca gives good treatment to accommodations and transportation) but not for some indispensable aspects of a destination (examples being cultural, natural and artistic attractions or events).

For this reason we have used WordNet in our feasibility analysis, and the decision as to which concepts to consider was informed by the definition of destination itself and by the table. For example, for alpine destinations we used the concepts of sport, landscape, nature, etc. We were thus able to simulate the nucleus of an ontology for tourist destinations.

Table 1. Classification of destinations based on their principal attractions

Type of destination	Main reasons for visiting	Well-known examples	Typical attractions found at the destination
Urban	Culture, art, architecture, shopping	Capital cities	Museums, historic buildings, shops
Beach/Sea	Relaxation, enjoyment, socializing, sports, night-life	Rimini, Ibiza, Miami	Beaches, organized activities, amusement parks, discos, bars, pubs
Alpine	Outdoor sports, landscape and environment, nature, traditional events and customs, folklore	Cortina, Chamonix, Aspen	Nature trails, views, ski trails and slopes, ski-lifts
Rural	Get back to nature, local traditions in agriculture and production	Tuscany, Provence	Local food producers and agritours, visits to farms and vineyards
Wellness	Health treatments, relaxation, diet and exercise programmes, stress relief	Fiuggi, Baden-Baden	Places equipped for health and therapeutic treatments, areas for complete relaxation, medium- and high-level accommodations facilities, fitness
Religious	Renewal or deepening of faith, symbolic value of the location, spiritual retreat and introspection, solitude	Lourdes, Fatima	Place of pilgrimage, religious practices and celebrations
Third World	Adventure, discovery of other cultures, understanding of tribal life (rites, traditions, lifestyle) anthropological investigation	Yemen, Madagascar	Cities, historic places, rites, customs, celebrations, guided tours, contact with non-western local cultures
Exotic and Exclusive	Beautiful scenery, isolated locations, far from tourist trek, status symbol and image	Maldives, Seychelles	Villages in traditional style but with all modern conveniences, privacy, untouched natural environments

3. The use of ontologies to define specialized models

3.1 The Quality Model Factory

In [9] we described a modular and scalable approach – the Quality Model Factory - to define specialized quality models identifying the specific features of tourist destination Web sites. Its goal was to introduce a systematic way to define a "personalized" evaluation framework. The use of modules derives from the application of reuse of artefacts [13] as a viable practice for definition of evaluation models. Scalability is obtained thanks to the adoption of a general conceptual framework, for Web site quality, the 7Loci meta-model (www.economia.unitn.it/etourism/pubblicazioni.asp).
This model introduces seven dimensions used to classify the numerous features of a Web site that can then be evaluated. The dimensions are *Identity*, *Content*, *Services*, *Location*, *Maintenance*, *Usability* and *Feasibility*.

The foundational procedure that serves as the starting point in developing a modular model for a given class of Web sites is outlined in the steps in table 2.

Table 2. Procedure for the quality model factory

```
{1ST PART: DEVELOPMENT OF COMMON AND SPECIALIZED MODULES}
IF no model for the class of sites currently exists
THEN    FOR each of the 7Loci dimensions pertinent to the project
        Identify the requirements common to all sites in the class and convert them
        into a question; add the question to the Common module;
        Identify the specific requirements for the type of site under evaluation and
        convert them into a question; add the question to the Specialized module;
ELSE    FOR each dimension of the 7Loci:
          FOR each question of the existing model
            IF the question is applied to the type of sites in its current form
            THEN Add the question to the Common module
            ELSE IF the question requires only a formal modification
                 THEN Modify the question and add it to the Specialized
                 module;
            IF the question is inapplicable to the type of site under evaluation
            THEN check whether there is an alternative question and add it to the
            Specialized module
{2ND PART: COMPLETION OF COMMON AND SPECIALIZED MODULES}
FOR each requirement for the type of site under evaluation
Identify the 7Loci dimension it refers to
IF no question exists for it in the Common or Specialized module
THEN IF the question regards all the sites in the class
       THEN Add a question to the Common module
       ELSE Add a question to the Specialized module.
```

The procedure has two parts; the first is the instantiation of the 7Loci model, reusing where possible a model already defined for one of the types of site in a class. This step is necessary in order to define the model using requirements as the starting point. In the second part the specific elements for the type of site are identified. When talking about the sites of tourist destinations, this means translating these elements into points (characteristics) within the evaluation model; for example, the unique features of the site for a seaside destination will be converted into points in the model used to evaluate that type of site, while features for another type of destination (religious, for example) may not contain those same points. In our previous work we looked at the aspects in table 1, which in general can be

found in classification schemes of the sites of a specific category. When describing the procedure, for the sake of simplicity we refer to "questions" to insert in the evaluation modules. In reality this is only one way of formulating the points or factors of the evaluation model; besides interrogatives (e.g., in boolean questions) they can also be described in declarative form.

In this paper we want to broaden the approach to be able to develop models for diverse types of sites, and such generalization is conceptually based on the use of ontologies. The steps of the procedure where ontologies can be used are shown in table 2.

The logical architecture of the Quality Model Factory is depicted in figure 2 (a Quality Factory to support "information quality" assessment is described in [4]), where the database "Web site classification" is substituted with a database containing the ontologies.

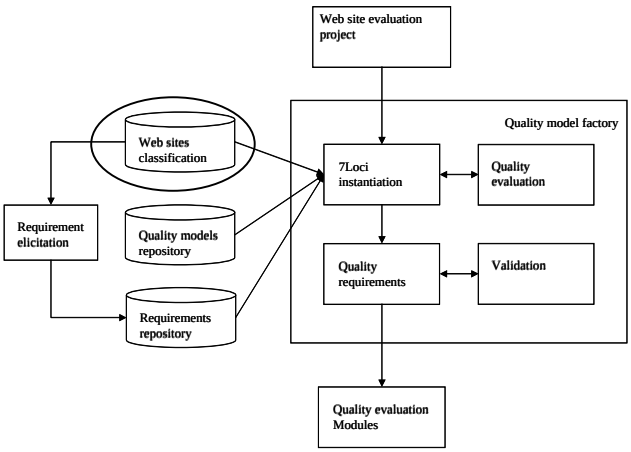


Figure 2. The Quality Model Factory

From among the different actors in the tourist sector present in the territory we focused on hotels and tourist destinations when studying our approach. For both categories tourism is the core business and the Web site is a strategic tool. For the tourist destination the Web site serves to promote and commercialize the products as well as to support the different actors.

3.2. Models for accommodations facilities

In a previous project we built a model for the comparative evaluation of the Web sites of about 200 hotels. The model consists of 18 questions: four for the dimension *Identity*, nine for *Content*, two for *Services* and one for *Localization*, *Management and Usability*. The model is a useful reference to check the results that can be obtained with the Quality Model Factory which is strengthened through the use of ontologies. In this case we have used the concepts related to *hotel* in WordNet. In addition, for the requirements of the owner and user, we have put together information emerging from the research conducted on the hotels [6]. In this case it meant developing a model more similar to that constructed with the contribution of experts, and also applicable to all hotels. Thus we did not intend to construct different models for different types of hotels.

3.3. Models for tourist destinations

Applying the Quality Model Factory approach to tourist destination Web sites without using ontologies produced a series of models, each containing about 100 factors that are a reference for evaluating and which can be produced with ontologies [9]. Moreover, we thus obtained important results for the adaption process, where ontologies are necessary to specialize the general-purpose models. Also emerging, in particular, was how the first two dimensions of the 7Loci meta-model depend more on the type of site (with an average of about 80% of specialized factors); about one third are specialized for *Services*, ten percent for *Usability*, *Maintenance* and *Localization* (related principally to the presence of different target tourist groups). Development of a model for the evaluation of Web site quality for tourist destinations is underway as part of the joint project IFITT/WTO (www.ifitt.org/)sito. The model foresees two levels of assessment: the first looks at general aspects and therefore at requirements that all tourist destination Web sites must satisfy; the second is called Strategy Based Full Web Site Evaluation. A limitation of the IFITT/WTO model is that there is a measurable gap between the two levels. Essentially the first level uses a general model for all destinations while the second must be defined ad hoc. We propose the use of ontologies to specialize the quality models in a systematic way.

4. Conclusions

The project is still underway, the results obtained thus far have revealed the following aspects:

- The dimensions of the 7Loci meta-model where the use of ontologies is more straightforward are

those which are more "semantic", thus *Content*, and *Services*; for these dimensions it is possible to use the hierarchies contained in the ontologies as check-lists to identify aspects for which the site must give information or support for services.

- For the dimension *Identity* it is necessary to integrate the ontologies with the aims of the site and be able to connect them to the concepts in the ontology that mainly contribute to the creation of the image of the organization; these concepts are usually specializations of "father" concepts; in the case of hotels, their specialization is for specific target markets, for example motor hotels.
- For all the dimensions of the 7Loci meta-model, the combined use of ontologies, of the standardized version accompanied by a list of the main aims of the site (no more than five elements: e.g., target, business functions, general links), by people with limited experience working in Web site quality (undergraduates students), made it possible to develop "draft" specialized models in a short time and which can be rapidly verified and completed by an expert. On the whole, the methodology proposed makes it possible to notably improve the efficiency of the process of defining specialized models.
- Substantial initial effort is required to identify ontologies for the different domains necessary to cover the tourist sector. Moreover, existing ontologies are heterogeneous with regard to the coverage of the domain they refer to. Nonetheless, they can be reused for numerous categories of operators and entities.
- Statistical analysis of the terms used in sites under analysis provides useful information for the choice of ontology, but principally to choose the concepts to use within the ontology to instantiate the quality models.
- Most existing ontologies are in English language; but it could be necessary to have ontologies in other languages.
- As for their implementation, most ontologies that can be used in the tourism context are written in DAML and some in OWL. This means that if we want to create environments to support the "Quality Model Factory" it is necessary to extract a version of at least two different types of file that even non-experts can understand.

10. References

[1]. Bachlechner D., D10 v0.2 Ontology Collection in view of an E-Tourism Portal, *E-Tourism Working Draft 5*, Oct. 2004, http://www.deri.at/research/projects/e-tourism/2004/d10/v0.2/20041005/

[2]. Barnes S. J., Vidgen R. T., WebQual: An Exploration of Web Site Quality, H. Mahrer (ed), *Proc. of the 8th European Conf. on Information Systems*, Vienna, July 3-5, 2000, pp. 298-305

[3]. Berners-Lee T., Hendler J., Lassila O., The Semantic Web, *Scientific American*, May 2001, 284(5): 34-43

[4]. Capiello C., Francalanci C., Pernici B., A rule based methodology to support information quality assessment and improvemnet, *Studies in Communication Sciences*, 4(2): 137-154, 2004

[5]. Falbo R.A., Guizzardi G., Duarte K.C., Natali A.C.C., Developing Software for and with Reuse: An Ontological Approach, *Proc. CSITeA'2002*, USA ACIS, 2002, pp. 311-316

[6]. Franch M., Martini U., Novi Inverardi P.L., Buffa F., Awareness and exploitation of the potential of the Web by SMTEs: The case of alpine hotels in Italy and France, Frew A.J. (ed.), Springer Verlag *LNCS 2722*: 318-327, 2005

[7]. Hüsemann B., Vossen G., Ontology-Driven Multimedia Object Management for Private Users – Overview and Research Issues, *AIS SIGSEMIS Bulletin*, 1 (1), April 2004

[8]. Martini U., Da luoghi a destinazioni turistiche. In Franch, M. (ed.), *Destination management: Governare in turismo tra locale e globale*, Turin, Giappichelli, pp. 67-111, 2002 (in Italian)

[9]. Mich L., Franch M., Martini U., A Modular Approach to Quality Evaluation of Tourist Destination Web Sites: The Quality Model Factory, Frew A.J. (ed.), Springer Verlag, *LNCS 2722*: 555-565, 2005

[10]. Mich L., Franch M., Novi Inverardi P., Marzani P., Choosing the "rightweight" model for Web site quality

evaluation, Frew A.J. (ed.), Springer Verlag, *LNCS 2722*: 334-337, 2003

[11]. Olsina L., Rossi G., Measuring Web Application Quality with WebQEM, *IEEE Multimedia* 9(4): 20-29, 2002

[12]. Pahl C., Casey M., Ontology support for Web Services Processes, *Proc. 9th ESEC and 11th SIGSOFT Symposium FSE*, 2003, pp. 208-216

[13]. Pressman R.S., *Software Engineering: A Practitioner's Approach*, 5th ed., McGraw-Hill, 2001

A Web Service for Hypermedia Role-Based Policies

Daniel Sanz, Ignacio Aedo, Paloma Díaz
Laboratorio DEI. Departamento de Informática
Universidad Carlos III de Madrid
{dsanz | aedo | pdp}@dei.inf.uc3m.es

Abstract

High level security is a key requirement in hypermedia/web applications. The systems opening to the Internet makes the research effort to move towards protecting the information transmission (i.e. SOAP messages, policy descriptions...), but little attention is paid to what the user can do with the system. Role-based access control (RBAC) allows to formulate the organization's resource access policy in a natural way, so a role-based access model for hypermedia will make it easier to integrate the security design with the rest of the system design. In service oriented architectures, an access policy service would allow to gather the management and deployment of the security policy in distributed and heterogeneous environments. This paper describes a role-based access control model for hypermedia called MARAH, and its implementation as a web service. An use case of the model in the design of the ARCE application is also discussed.

1. Introduction

The industry and research community trend towards the openness and interoperability among systems is very clear, what makes very important the adequate resource protection. The balance between service providing and asset protection becomes an strategic issue, so the security design has to be more than an implementation/deployment aspect: security is a first-class element whose requirements should be taken into account throughout all development process. This design integration would help to face the security analysis and design [4], but in order to achieve it with success, a formal basis gathering the relevant concepts in the protection domain is required, taking the form of formal models. In the highest abstraction level, access control is a security service that allows to regulate how users interact with the system.

Hypermedia/web applications are implemented in a variety of platforms, where the information is probably distributed and heterogeneous, and the access context can be very different each time. There is a clear need for a model that considers these issues and makes it possible to express the access policies in a flexible way. From an architectural point of view, it is recommended to keep some separation between the security controls and the controlled resources themselves, what makes the Web Services paradigm a perfect deployment platform to test our ideas.

MARAH is a role-based access model for hypermedia applications, that works with concepts of the hypermedia domain (i.e. contents or links), and is based on the RBAC model. RBAC articulates the access policy around the role concept: instead of granting permissions directly to system users, permissions are assigned to roles, and users are assigned to the roles they can play, acquiring its permissions. The role represents job functions, responsibilities or qualifications in the context of some organization, and groups both the permissions and the users allowed to use them.

The use of roles and proper hypermedia abstractions to express the access rights makes it possible to integrate MARAH in a development method, due to designers can decide about what the access policy is while designing other system aspects, such as navigation or content layout. This facilitates the policy definition, because now access control is not performed on the application code in terms of classes/objects, database tuples or other paradigm, but protected resources are the elements of a hypermedia design, that have a correspondence with one or more systems responsible of the information generation.

MARAH is available as Web Service, with the goal of providing the access policy information in a concrete domain. The implementation covers the core RBAC standard [2], and does not assume any specific hypermedia model, so it can be used from any hypermedia implementation. This work is organized as follows: section 2 presents a reduced version of the model, based on core RBAC specification. Section 3 describes the implementation and architecture of the MARAH web service. Section 4 introduces an use case in the ARCE application, and finally section 5 discusses some conclusions and future works.

2. The MARAH model

Initially MARAH was defined as an unique model, specifically designed for a hypermedia reference model called Labyrinth[6, 7]. After some implementation experiences [8, 11] and the creation of the RBAC ANSI standard [2], it is being restructured for a twofold purpose: to define the model features in an incremental way, as the RBAC family of models do, and to be independent of the underlying hypermedia model, and so applicable to any hypermedia architecture. We try to make more flexible the use of the model in a variety of situations, and to face its transformation to an independent service. We are working on the core of the model (Core MARAH in this paper), that, starting from the core of RBAC, includes the very basic elements for a hypermedia access control model.

Next subsection introduces the core RBAC model, as previous requirement to define the core MARAH model.

2.1. The core RBAC model

The core RBAC is one of the four model components defined in the RBAC reference model, and "defines a minimum collection of RBAC elements, element sets and relations in order to completely achieve a Role-Based Access Control system" [2]. The elements of the core RBAC model are:

- U, R, OPS, O
  Sets of Users, Roles, Operations and Objects respectively.
- $UA \subseteq R \times U$
  A many-to-many mapping user-to-role assignment relation.
- assigned_users : $R \rightarrow 2^U$
  The mapping of a role r onto a set of users allowed to hold it. Formally, assigned_users$(r) = \{u \in U \mid (u, r) \in UA\}$.
- $P \subseteq 2^{OPS \times O}$
  The set of permissions. A permission represents an approval to execute an operation on a protected object.
- $PA \subseteq P \times R$
  A many-to-many relationship mapping the permission-to-role assignment relation.
- assigned_permissions : $R \rightarrow 2^P$
  The mapping of a role r onto a set of permissions. Formally, assigned_permissions$(r) = \{p \in P \mid (p, r) \in PA\}$.
- Op : $P \rightarrow OPS$
  The permission to operation mapping, which gives the set of operations associated with permission p.
- Ob : $P \rightarrow O$
  The permission to object mapping, which gives the set of objects associated with permission p.
- S
  The set of sessions. A session represents an user interaction period, where the user activates some subset of the roles she is assigned to. Each user can establish one or more sessions.
- session_users : $S \rightarrow U$
  The mapping of session s onto the corresponding user.
- session_roles : $S \rightarrow 2^R$
  The mapping of session s onto a set of roles. Formally, session_roles$(s) \subseteq \{r \in R \mid (\text{session_users}(s), r) \in UA\}$
- avail_session_permissions : $S \rightarrow 2^P$
  The permissions available to the user in a session. Formally, avail_session_permissions$(s) = \bigcup_{r \in \text{session_roles(s)}} \text{assigned_permissions}(r)$

In addition to the sets, relations and functions, the RBAC standard also defines the system and administrative functional specification for the core model. Administrative functions allow to create and maintain the RBAC sets and relations, administrative review functions allow to perform administrative queries, and system functions allow to create and manage user sessions and make access control decisions.

2.2. The core MARAH model

The core of MARAH is based on a hypermedia metamodel that assumes two fundamental elements of this discipline: the distinction between container and content, and the information linking. As in RBAC standard, to apply MARAH in a concrete system, a mapping from the access model components to the system components has to be provided. Due to MARAH is an access control model for hypermedia, this mapping should not be very complex, as MARAH provides the proper abstractions representing protected objects. Starting from the core RBAC specification, the Core MARAH model provides three contributions.

The first of these contributions is the definition of the protected objects. The hypermedia metamodel assumed by MARAH defines four object types[1] that can be identified in most hypermedia models:

Nodes abstract information containers, they have whole meaning in the application context. Represent presentation units.

[1] These object types reflect the application information structure, regardless how the information is modeled (i.e. OO model), stored or generated to give place to such structure.

Contents individual pieces of information that are included in the nodes. They can be simple (e.g. text) or complex (e.g. video).

Anchors represent areas or fragments in the node or content representation space (i.e. characters, pixels, rows, seconds...).

Links represent information connections, by means of relationships between a set of source and a set of target anchors.

Nodes and contents represent the system information, so they are the basis on which privileges are established. Anchors and links are used to represent relationships between information elements, so its definition is somehow tied to the elements in which are defined, and thus its access rights have to be derived from the permissions of these elements.

The second is the addition of relationships among objects. Core MARAH considers two relation types in the object set:

Location reflect the spatial/temporal placements among objects. For instance, the inclusion of contents into nodes or the definition of anchors.

Linking represent associative connections between nodes and/or contents.

As explained later, these relationships are used to propagate access rights in a variety of situations, so the administration cost is reduced due to there is no need to specify the permissions for each object.

The third contribution is the addition of one abstraction level on top of the system operations, reducing the number of permissions assignments by means of a classification of operations. The classification groups the operations and allows to easily set permissions involving several operations.

Core MARAH is an extension of Core RBAC standard that defines the object types[2], new elements and the relationships between the RBAC standard model, as shown in figure 1:

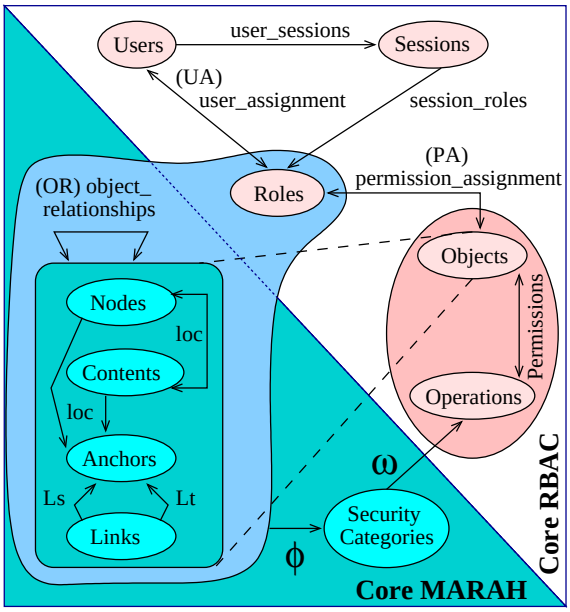


Figure 1. Core MARAH model and Core RBAC model

- $O = N \cup C \cup A \cup L$
  Sets of nodes (N), contents (C), anchors (A) and links (L) respectively.
- $OR = \{\succ_{loc}, \succ_{Ls}, \succ_{Lt}\}$
  Set of object relationships. Core MARAH defines three relationship types:

  Location. $\succ_{loc} \subseteq O \times O$. If a $\succ_{loc}$ b, there is a spatial/temporal relationship from object a to object b. For instance, if anchor a is defined to start in the third second of video b, then there is a location relation from the anchor to the video.

  Source linking. $\succ_{Ls} \subseteq A \times L$. If a $\succ_{Ls}$ b, the anchor a is an origin of the link b.

  Target linking. $\succ_{Lt} \subseteq A \times L$. If a $\succ_{Lt}$ b, the anchor a is a target of the link b.

- OPS
  Set of operations. Core MARAH only considers generic operations for system use, such as "seeNode" or "traverseLink". Like the object set, the operations are provided by the underlying hypermedia model.
- $SC = \{B, E\}$
  Set of security categories, with two elements. Each category represent abstract manipulation types that exist in any hypermedia system. The value B represents "browsing" and allows to see the element, and the value E means "editing" and allows to modify or create an element. A partial order is defined in SC such as $E > B$, since the edition manipulation includes the viewing manipulation.
- $\omega : OPS \rightarrow SC$
  Classification of operations function, that relates each operation with the manipulation type that its execution requires, for instance, $\omega(\text{getLinksToTarget}()) = B$.
- $\phi : R \times \{N \cup C\} \rightarrow SC$
  The clearance function returns the highest manipulation ability that a role can do on an object. This value is set by the security officer for contents and nodes,

[2] Due to its generality, the RBAC standard leaves open this question.

and involves the creation of a set of permissions in P for the object and the operations classified with a lower or equal category than the one specified. Formally, if administrator sets the following clearance $\phi(r,o) = E$, then $\forall op \in OPS \mid \omega(op) \leq E \Rightarrow P = P \cup (op,o), PA = PA \cup \{(op,o),r\}$.

- $\prod : S \times O \rightarrow SC \cup 0$
  The authorization rule returns the highest manipulation ability available in a session for a given object. The value depends on the session active roles, the type and relationships of the involved object. The function is defined as follows:
  - For nodes and contents, returns the maximum of the session permissions:
    $\prod(s,o) =$
    $\max(0 \cup \{ \bigcup_{p \in \text{ASP}(s), Ob(p)=o} \omega(Op(p))\})$
    $(\text{ASP}(s) = \text{avail\_session\_permissions}(s))$. For the maximum, a total order is defined such as $E > B > 0$. The value of 0 indicates the absence of permissions for the object in the session[3].
  - Anchors take directly the permission granted to the object on which are located.
    $\prod(s,a) = \prod(s,o) \mid a \underset{loc}{\succ} o$
  - For links, "B" access is allowed if there is access to at least one anchor in each link end. "E" access is allowed if all anchors have editing permissions:
    $$\prod(s,l) = \begin{cases} B & \text{if } \exists a_1, a_2 \mid a_1 \underset{Ls}{\succ} l, a_2 \underset{Lt}{\succ} l, \\ & \prod(if, a_1) = \prod(s, a_2) = B \\ E & \text{si } \forall a_i, a_j \mid a_1 \underset{Ls}{\succ} l, a_j \underset{Lt}{\succ} l, \\ & \prod(s, a_i) = \prod(s, a_j) = E \\ 0 & \text{otherwise} \end{cases}$$
- $\theta : OPS \times O \times S \rightarrow \{0,1\}$
  The transition function checks if an operation can be performed on an object from a given session.
  $$\theta(op,o,s) = \begin{cases} 1 & \text{if } \omega(op) \leq \prod(s,o), \\ 0 & \text{otherwise} \end{cases}$$

The Core MARAH model defines also a set of administrative and system functions, in order to create, maintain and query the different sets and relations, as well as creating user sessions and taking access control decisions.

3. Core MARAH Web Service

Core MARAH is being implemented as web service. It acts as *policy decision point* (PDP), independent of the *policy enforcement point* (PEP) where the actions according to the PDP take place, typically in systems responsible of content generation, integration and/or delivery. Previous implementations of MARAH were integrated in the content server [8], making no distinctions between the PDP and PEP, and constraining its use to only one website. The MARAH service offers a RBAC standard implementation that is reusable by several applications within the organization. As figure 2 shows, the differences between MARAH

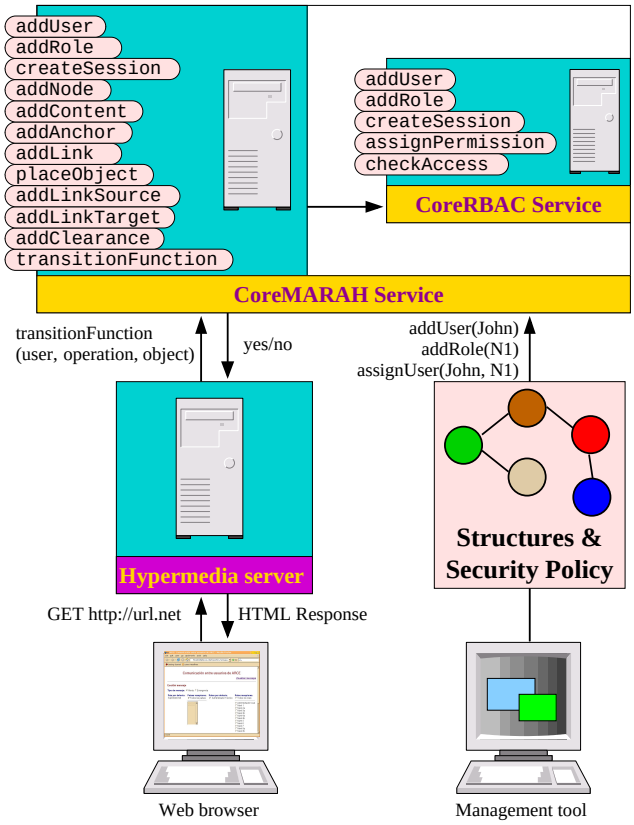


Figure 2. Core MARAH service architecture

and RBAC standard have led to a broad service split in two smaller cooperating subservices. This cooperation gathers the transformation rules from MARAH components to RBAC components described in previous section, in terms of service operation execution, highlighting the fact that Core MARAH is an additive extension of Core RBAC. From the point of view of the service clients, whether administrators or systems implementing PEPs, this separation is completely hidden. The service usage is very simple: first, access policy is notified to the service. Then, the service is requested for access control decisions involving operations, objects and roles previously defined. Here the term access policy means the definition of the set of objects and its relations, roles and clearances. It is important to note that the adoption of the web service paradigm and the cre-

[3]The model does not provide an interpretation for this situation.

ation of a policy service separated from the PEP introduces new challenges, due to there is a need to protect the information during transmission between PEP and PDP. Due to the focus of this work is the deployment of the abstract core access control model, and taking into account the advances to come in the web services security area [9], issues related to low level security in the service usage (i.e. identification, authorization, end-to-end encryption,...) have not been considered yet.

The Core RBAC web service implements the Core RBAC administrative and system functional specifications, providing operations for:

- Adding/removing users and roles,
- Assigning/deassigning users to roles, and permissions to roles,
- Creating/deleting sessions for an user and activating/deactivating some roles,
- Checking the access.

In order to provide a general service, all elements (roles, users, permissions and operations) are identified by `string` tokens, whilst the service returns information using primitive types such as string lists or boolean values. Figure 3 shows part of the web service definition (WSDL) for Core RBAC service.

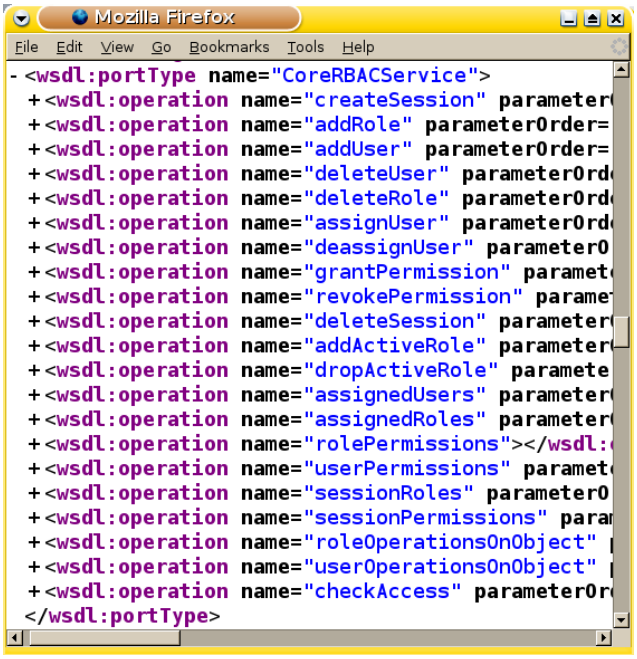


Figure 3. WSDL fragment for the Core RBAC service

The Core MARAH service acts as client of the Core RBAC service, and implements administrative and system functions, that, in addition to the functionality provided by Core RBAC, allow to:

- Creating/deleting object instances, for each one of the four types,
- Creating/deleting object relationships (location, source linking and target linking),
- Granting/revoking clearances to roles (for nodes and/or contents),
- Checking the access, by the transition function (θ).

The semantics for permission propagation, mainly gathered by the authorization rule ($\prod$) are the most important part of the Core MARAH implementation, due to represents a hypermedia related abstraction layer on the objects not present in the Core RBAC service.

Figure 4 shows part of the WSDL definition for the Core MARAH service. The implementation of some of its operations make use of the Core RBAC service, some of them being simply wrappers for the RBAC service operations. Core MARAH service also uses string based primitive types.

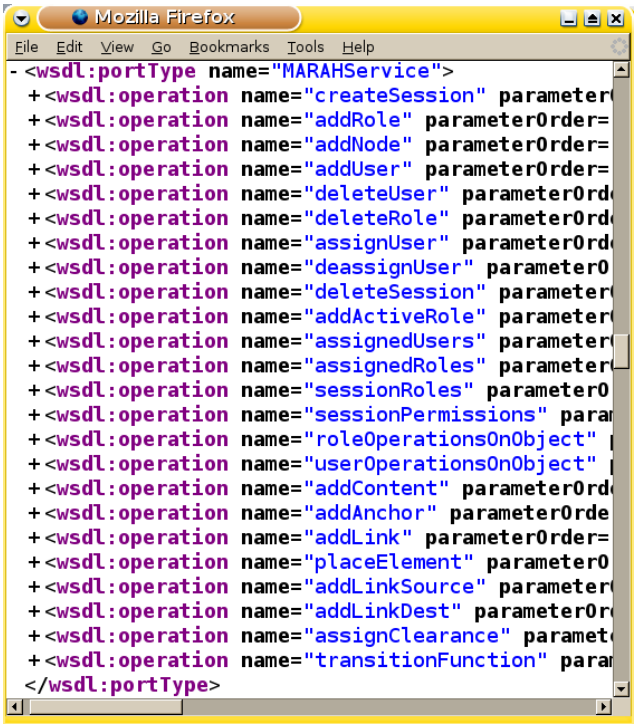


Figure 4. WSDL fragment for the Core MARAH service

Both Core RBAC and Core MARAH services are being developed using Java, and made available as web service using the Axis software, from the Apache project (http://ws.apache.org).

4. Use case: the ARCE system

The Core MARAH service is being probed in an use case of the ARCE system (networked application for emergency situations), which goal is to facilitate the efficient aid management in emergency situations among Latin-American countries [1]. This application uses RBAC mechanisms to establish who can create emergencies, ask for resources and negotiate contributions for the opened emergencies, but the access policy is hardwired into the code, and so the access decision is taken each time using dedicated lines of code (PDP and PEP are tightly coupled). In order to introduce MARAH here, we have started to completely decouple the access decision in the part of the system related to messages. This section briefly describes our first steps towards this direction.

One of the ARCE functions is to promote the communication between users, and so a messaging mechanism was designed, similar to the email, but the receiver of the message is a role, and not a concrete user. There is an information flow policy such that one given sender can write messages only to a concrete set of target roles. There are roles representing different job functions and organizations in the Latin-american Association of Governmental Organisms of Civil Defence and Protection. Roles N1, N2, N3, N4, N5, N6, N7 and N8 are message senders. Each one of them has different qualifications, manages different information views and they do not make a role hierarchy (e.g. N1 does not take all N2's permissions). For the sake of clarity, we will describe a subset of the information flow policy.

- Role N2 sends messages to N2 receivers, with mandatory copy to N1. The rest of receivers can be selected by the sender when composing the message.
- Role N3 sends message to N3 users, with mandatory copy to N1. The rest of receivers can be selected by the sender, excepting N8, that can't receive messages from N3.

The messaging subsystem has three pages: the message composition page, the message list page (provides a table with message subjects) and the message view page (showing one message).

It is important to note that the security model itself is not directly concerned with other important hypermedia aspects such as navigation or presentation. These issues are supposed to be handled by the hypermedia reference model, or somehow considered in the design method. In our case, the system implementation is responsible for taking the right action (PEP) according to the response of the PDP, including presentation issues. The implementation is also responsible of notifying the MARAH service when new elements/relationships are created. The system has been modeled as follows:

- Nodes: the message composition, message list and message view are nodes. For each new composed message, a new node is generated representing it.
- Contents: Each form element in the message composition page is a content. In the message list, there is a table which rows representing links to messages. In the message view page, the message title and text are also contents. Each new message generates contents for the text and title.
- Anchors and links: from the message composition the user can navigate forwards and backwards to the message list. In this page, there is a link for each written message pointing to the corresponding message view node.
- Clearances: The message composition and list nodes have the "B" category for everybody. The message composition form has checkboxes in order to select the receiver roles. These checkboxes have different permissions according to the information flow policy:
  - If the sender *must* send the message to the receiver, the checkbox has a "B" clearance, stating that the checkbox value can't be modified
  - If the sender *can choose* if the receiver will read the message or not, then the checkbox has a "E" clearance
  - If the sender *can not* send the message to a receiver, there is no permission for the checkbox

According to this modeling, table 1 shows the clearances for the contents of message composition node, for roles N2 and N3 (the symbol $\emptyset$ indicates that no clearance is assigned). Each checkbox is named using the destination role, but this is a convention. Figure 5 shows the aspect of the generated pages in conformance with the policy for roles N2 and N3, where the non-editable checkboxes appear disabled. If some receiver must be notified, the corresponding checkbox appears marked and disabled.

The new version of the messaging subsystem has been implemented in PHP due to this language offers SOAP functions that make it easy the coding of web service clients. The web server chosen to host the PHP pages is Apache. Nodes and contents are PHP objects that provide their information through the `show()` method, that

	N2	N3
checkbox_N1	B	B
checkbox_N2	B	E
checkbox_N3	E	B
checkbox_N4	E	E
checkbox_N5	E	E
checkbox_N6	E	E
checkbox_N7	E	E
checkbox_N8	E	∅

Table 1. Values of the clearance function for roles N2 and N3 in the ARCE messaging system

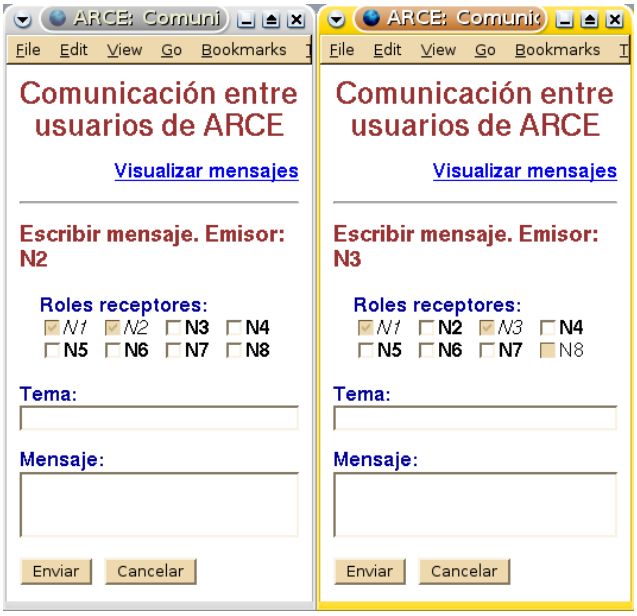


Figure 5. Views of message composition page for roles N2 and N3

includes a call to the Core MARAH service transition function. The PDP queries are performed by the class pdp.php, whose objects are built by providing the URL of the WSDL description of the Core MARAH service, as shown in figure 6.

```php
<?php
class pdp {
 private $client;
 public function __construct($wsdl) {
   $this->client = new SoapClient($wsdl);
 }
 public function safe($user, $object, $operation) {
   return $client->transitionFunction($user,
                        $object, $operation);
}
}
```

Figure 6. Core MARAH web service invocation from PHP

Each node and content in Core MARAH corresponds to a PHP object, but nodes keep references to the objects representing their contents (and thus implementing the $\succ_{loc}$ relation), as shown in figure 7, where a generic node code is shown. Using this mechanism, a PHP node will perform several queries to the PDP during its dynamic generation. The process for the generation of the message composition node (see figure 5) is as follows: once policy is notified to PDP, an authenticated user tries to visualize the page, so PHP calls `show()` on the PHP object implementing the page. First, the node checks access for itself, using context information that includes the user name and session identifier. After querying the PDP and checking the transition function value for the "B" category, the node starts to show its own contents, according to its own source code.

```php
<?php
class node {
    private $pdp;
    private $id;
    private $contents;
    public function __construct($pdp, $id) {
        $this->pdp = $pdp;
        $this->id = $id;
    }
    public function placeContentNode($content) {
        $this->contents[] = $content;
    }
    public function show($user) {
        if ($this->pdp->safe($user, $this->id, "browse")) {
            $this->beginWrap();
            foreach ($this->contents as $content) {
                $content->show($user);
            }
            $this->endWrap();
        }
    }
```

Figure 7. Generic code for a node

Each content to show is a PHP object, that is notified to show itself by the `show()` method. In order to show its information, the transition function has to return 1 for the content itself, so the PDP is queried again. When the content includes anchors or other contents, the process is repeated again, so each HTML element sent to the client has been filtered by the access control mechanism, and so it is an allowed element. In this way, a given sender can not select roles for which sending a message is prohibited, because the contents that allow this action are skipped or otherwise disabled from the page.

Once the sender writes the message title and body, selects the receivers and sends the message, the system does the following:

- Generate a new node n representing it, including the message title and body in its contents. Define an anchor t and include it in the node.
- Add the message title content in the message list page, creating an anchor s for it
- Create a new link, whose source is the s anchor, and the target is t.
- Set the clearances corresponding to the new node for the selected roles, using the checkbox information. If a role N_i is marked, then $\phi(N_i, n) = B$.

Now we illustrate how links are skipped in the message list node. When processing this node, each table row represents a link to a message. In order to get access to a link, one of its sources and targets must be accessible. Since the target anchor takes the permission from the node in which is defined, the whole link has the permission of the destination. This implies that if the message is denied to some role, the link to it is too denied, so the PHP page will not add the corresponding row to the table. So, it can be said that there are one or more PEP for each object, but the PDP answers are provided by Core MARAH service.

5 Conclusions and future work

This work presents a role-based access control model for hypermedia systems, and its deployment as a web service, in response to the need of separating the access control decision from the policy enforcement. Moreover, a prototype that uses the web service has been built, in the ARCE application context. The model is deliberately simple, because its design criteria are based on the modularity, flexibility and minimalism, in similar philosophy as the used in the core RBAC design.

There are several future work directions. Firstly, the access model will be extended by adding several layers in order to add more advanced features. The first one will presumably be the Hierarchical MARAH, that will be based on Hierarchical RBAC and will extend it by providing new roles and objects relationships. This will allow to create richer structures where the permission inheritance will reduce even more the access rights management. Other foreseen extension are the inclusion of context parameters [5], specially useful in open environments because they provide more information to make access control decisions (i.e. location from where the access is requested). We also plan to add temporal constraints [10], that allows to enable or disable the use of roles or permissions assignments for a given time interval or period, as well as limiting the duration of the sessions or permissions. Since we foresee a very rich set of additions, the formalization of the hypermedia metamodel assumed by MARAH, with extensibility in mind, becomes a fundamental issue. Moreover, this formalization would allow to provide bindings between the MARAH abstractions and the modeling elements provided by other hypermedia reference models, thus making able to integrate MARAH in different design methods.

Regarding the service, we plan to study how to apply the service in decentralized architectures where there are several PDP responsible of keeping parts of the enterprise-wide access policy. The PDPs can also be separated to some extent and be transformed into another service, able to generate an adequate version of the content according to the user, for instance using SMIL (http://www.w3.org/AudioVideo/).

Moreover, we will consider the integration with web services security standards, particularly WS-Policy [3], for environments where contents are provided by web services that describes and attach their access policy as part of the service description, maybe in different formats such as XACML or SAML. These policies would be used by one or more PDPs that gather and interpret them in order to provide an access control decision.

6 Acknowledgements

This work is part of the ARCE++ project (TSI2004-03394) funded by the Spanish Ministry of Science and Education (MEC) and a cooperation agreement between Universidad Carlos III de Madrid and Direccin General de Proteccin Civil y Emergencias (Ministry of the Interior).

References

[1] I. Aedo, P. Díaz, C. Fernández, and J. Castro. Supporting efficient multinational disaster response through a web-based system. *E-Gov Conference. Aix-en-Provence (France).*, pages 215–222, Sept. 2002.

[2] ANSI. Incits 359 2004. american national standard for information technology. role-based access control, 2004.

[3] S. Bajaj. Web services policy framework. http://www-128.ibm.com/developerworks/ library/specification/ws-polfram/index.html, 2004.

[4] G. Brose, M. Koch, and K. Lohr. Integrating security policy design into the software development process. Technical Report B-01-06, Freie Universitat, Berlin, Nov. 2001.

[5] J. Crampton. Specifying and enforcing constraints in role-based access control. In *Proceedings of SACMAT 2003, Como, Italy*, pages 43–50, 2003.

[6] P. Díaz, I. Aedo, and F. Panetsos. Labyrinth, an abstract model for hypermedia applications. description of its static components. *Information Systems*, 22(8):447–464, 1997.

[7] P. Díaz, I. Aedo, and F. Panetsos. Modelling the dynamic behavior of hypermedia applications. *IEEE Transactions on Software Engineering*, 27(6):550–572, June 2001.

[8] P. Díaz, D. Sanz, and I. Aedo. Marah: an rbac model and its integration in a web server. In K. Morgan and M. Spector, editors, *The Internet Society: Advances in Learning, Commerce and Security*, pages 243–252. WITPress, 2004.

[9] W. A. W. Group. Web services architecture. http://www.w3.org/tr/2004/note-ws-arch-20040211/, 2004.

[10] J. Joshi, E. Bertino, U. Latif, and A. Ghafoor. Generalized temporal role based access control model (gtrbac) (part i) - specification and modeling. https://www.cerias.purdue.edu/infosec/ bibtex_archive/archive/2001-47.pdf, 2001.

[11] S. Montero, I. Aedo, and P. Díaz. Generation of personalized web courses using rbac: Courba, a practical experience. *AH 2002*, pages 419–423, May 2002.

Developing an Alarm Manager Based on Web Services*

Daniela Leal Musa, Marcel Weschenfelder, José Palazzo Moreira de Oliveira
Instituto de Informática - UFRGS
{musa, mweschenfelder, palazzo@inf.ufrgs.br}

Abstract

This paper describes an alarm manager aimed at monitoring students' activities in online courses over the Internet. The goal is to show an alarm manager to give more support to distance teaching, and help the teacher to have a thorough follow-up of distance students, thus minimizing the teacher's task of looking for information in long reports or complex graphs. Besides, the alarm manager also helps students during their study sessions, sending extra material, help when it considers necessary. The alarm manager proposed here was incorporated into the distance-teaching environment known as Claroline. The propodsed solution is an approach to identify the necessary requirements for alarms integration with web-based distance teaching environments situations.

1. Introduction

Today, most of the web-based distance learning environments have a great concern with the management of their courses, the display of instructional material and the availability of communication tools that foster the interaction between students and teachers. These environments lack mechanisms that would enable the teacher to perform a more complete and comprehensive follow-up of student's activities during courses. This could happen by the identification of the students' knowledge level and their learning pace.

In distance learning courses, the teacher's presence is not as effective as it is in presential courses. The evaluation of students' individual efforts is a great reason of teacher's concerns, especially when a great number of participants are involved in the course. Obtaining results that enable the assessment of the class as a whole is another worry, as in distance learning the teacher is not present to evaluate this aspect.

Usually, the process of student's evaluation in web-based courses is performed only through the analysis of information collected through tests, exercises or extensive reports that contain the log of pages accessed during the course. That is the reason why counting on appropriate tools to monitor what each student is performing is very important: problematic situations can be detected and the teacher is warned about each student's behavior. The teacher would have access to the particular features of each student, and would be able to customize classes to each one, making a closer follow-up and guiding the student during study sessions.

In some web-based teaching environments, Artificial Intelligence agents assess the student's behavior. An agent monitors the processes of interaction between students and the e-learning system, aiming at identifying the learning process of each student [1]. The higher the amount of information about each student, more suitable will be the agents' help. In the majority of teaching environments that use student's behavior monitoring techniques, the required data about student's data are extracted from an analysis of the distance learning server logs. Such records can provide the behavior of each student in what concerns files access (pages), which are stored in the server. However, it is necessary to perform several searches in these records to obtain information that are important and that can help to discover the student's behavior. A system of alarms can be used to perform such searches and warn the teacher about what is going on with some of them.

The goal of the present article is to show an alarm manager to give more support to distance teaching, and help the teacher to have a thorough follow-up of distance students, thus minimizing the teacher's task of looking for information in long reports or complex graphs. Besides, the alarm manager also helps students during their study sessions, sending extra material, notes or help when it considers necessary.

* This work was partially supported by grants from CNPq and CAPES.

The alarm manager proposed here was incorporated into the distance-teaching environment known as Claroline[2]. It was a way of identifying the necessary requirements for its integration with web-based distance teaching environments.

The present paper is organized as follows. Section 2 shows some studies that have been developed for students follow-up in web-based distance learning environments. Section 3 shows the alarm manager proposed, and Section 4 describes the integration with the distance-teaching environment named Claroline and the alarm manager. Section 5 brings some conclusions and future works.

2. Related works

In general, the student's behavior is assessed by using the agents' technology developed by Artificial Intelligence. Otsuka et al's [3] model is composed of interface agents. These agents learn through observation and monitoring of user's actions, they work as personal assistants, collaborating with the user and with other agents in the accomplishment of certain tasks. The model proposed by Otsuka is composed of three modules: (1) follow-up module, which tracks the student's interaction within the environment and generates reports about their participation; (2) module of behaviour analysis, which selects and presents data that can help in the analysis of student's performance in a specific activity; (3) validation module, which is responsible for the construction of the student's profile. Up to the present moment, no result has been shown about the efficacy of the model proposed, or if it has already been implemented or if it is under use.

The conBa system (CONcept Based) [4] is composed of interface agents able of monitoring and advising students during navigation in the instructional material. The system is composed of two main agents, the professor and the tutor. The professor agent offers the interface to the human-professors, who perform the following activities: (1) manages the course instructional material; (2) discovers, through the tutor agent, which topics the students are learning; (3) modifies the student's profile, informing the tutor agent that a given topic can be considered as acquired. The tutor agent keeps a description of the pages the student accessed, which are considered studied topics. These agents do not take decisions, and do not act in the student's favor. They only generate suggestions based on pages already visited by students and on the structure of the online course. Agents do not take into consideration the student's previous knowledge nor their style and learning pace. It would be interesting if the agent could indicate extra material when student's needs were identified, and not simply when the student reaches the module's end or accesses a specific page.

Some environments propose the use of web-server log files to collect students' and course's information. Niegemann [5] presents the EDASEQ tool (Exploratory Data Analysis for Sequential Data) to help in the analysis of student's navigation history. The tool tracks the web-server log file, converts data to an Excel file and generates graphs about the student's navigation. However, a human analysis is necessary to retrieve important information from these graphics. According to Niegemann, the tools were not used with real navigation data; fictitious data were produced to test the tool's functionality.

The monitoring model presented by Peled [6] is composed of a filter that extracts information from the information log considered not necessary, like the navigator used, the accesses generated with double clicks, accesses of teachers or course administrative personnel, administrative tasks, login, etc. Data resulting from these filters remain in text type files. This system was tested in a course with 200 students, however no tool had been developed to make visualization of these information easier. An exhaustive analysis in texts generated is necessary, making the process unfeasible.

Current studies have showed that the majority of techniques used offer final reports with access data and, in general, a human analysis is performed to collect important information from these reports. The teacher should search each student's data but is not aware of what is happening during the learning process. There is a need for other alternatives that would allow obtaining information about the student and the use of this information in the evaluation process.

3. Alarm system

A solution for the problem presented in section 2 is the use of an alarm system that monitors the processes of student's interaction with the course environment and generates a navigation history. The alarm system makes an analysis of this history. When certain situations are detected, the teacher is warned about what is going on with some students. Thus, the system is able to discover when a student stops at a certain point of the course, meaning he may not have understood the topic approached. Data from this analysis can also be used to help students in their study sessions, by stimulating and indicating extra material, exercises, etc. The coordinator of the instructional material can also benefit from an alarm system, because information of pages accessed and other about possible problems in the site structure may be sent to the coordinator. Figure 1 shows the architecture of the Alarm manager.

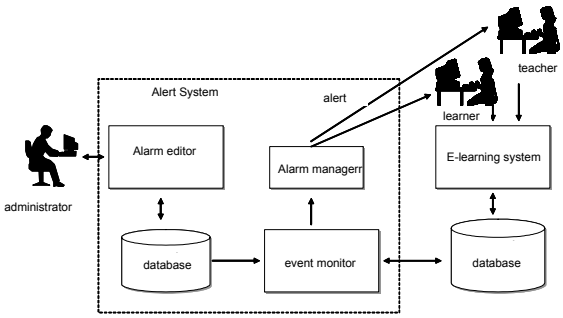


Fig.1. Alarm system architecture

The alarm system is composed of the following modules: alarm editor, events monitor, monitoring module, database, Log and alarm manager, which we present in details in the following sections.

3.1 Alarms creation and edition

Teachers or administrators use the alarm editor to define alarms used in the follow-up of student's activities. In this environment, each teacher can create alarms for their courses or select alarms that were already defined by other teachers. Figure 2 shows the user's interface for the definition of alarms in the current implementation of the system.

An alarm is composed of: (a) a triggering event, (b) one or more conditions that must be verified, and (c) an action that must be performed. Events are indicators of different situations, they can be two types: (a) environment's events, which are events the teacher or the student generates during their interaction with the teaching environment. Examples of this type of event are the student's *logon*, the end of some exercise or access to a certain page; (b) temporal events: some situations must be checked in a certain period of time, once a week, for example. This type of event depends on time and not on some situation that happened because of an interaction between the user and the environment.

An alarm is represented by ECA (event-condition-action) rules, that is, when an event takes place, a condition is checked and if it is true the action is performed. Rules are expressed as follows:

On event IF <condition> THEN<action>

When the alarm is created, the teacher selects the alarm-triggering event. In the current version of the system, the teacher cannot include a new event, and only chooses the events from a list of elements pre-defined by the alarm system.

Another information that must be informed at the moment of an alarm creation is the condition that will be tested at the moment the event is triggered. Today, the condition is informed to the system in the form of SQL queries, thus requiring the teacher has notions on the language and on the structure of the environment's data model. A solution that exports the teaching environment logic structure to the teacher and another form of the conditions representation is under study, so that the teacher can create his or her own alarms with no knowledge on SQL language.

The alarm action must be informed at the moment the alarm is created, and it can be a message in the navigator or in the e-mail. The message receiver, who can be the teacher, the student or the course administrator, must be informed. The type of action and the receiver are selected from a list previously defined, and it is not possible to insert a new action or receiver.

Data informed for the creation of an alarm are stored in the database shown in Figure 1. It is possible to make new searches, insertion and deletion of alarms through the Alarm editor. Any other teacher who participates in the system can use an alarm that has already been defined by a teacher.

3.2 Monitoring

The alarm system has a monitor of events that happen in the teaching environment database. The monitor consists basically of an interpreter of SQL commands. It receives an SQL command as parameter (Insert, Delete, Update) that will be performed in the teaching environment. The alarm server receives a warning from the monitor of events informing that some important event has taken place. After the warning, the alarm server verifies the conditions associated to that event. If the condition is true, the action indicated in the database of the alarm system is performed.

The alarm manager also stores the history of all alarms generated so that it is possible to perform statistics about the student's performance during the course.

4 Integration of the alarm system into the Claroline system

The alarm system we presented in the previous section was integrated into a distance-teaching environment so that the main requirements of integration of alarm systems into distance teaching environments could be

identified. This section describes this integration in further detail. The alarm system was integrated into the Claroline environment, which is distance-teaching software developed at the UCL (*Université Catholique de Louvain*) and is open code. Currently, 500 students and 20 professors of the Computer Science Institute of Universidade Federal do Rio Grande do Sul are using the software. Professors use the environment to make instructional material, exercises, and tasks schedules available, as a complement to presential classes.

Claroline was chosen for special reasons. First, because the environment adopts the policy of GPL open code software, a very important requirement in our work. Second, Claroline was developed in PHP language and MySQL database, the same used in the implementation of the alarm system, what made the integration between systems easier. The environment's source code is easy to understand its structure has environment variables and functions libraries, what helps the addition of other software modules.

4.1 Monitoring

There is a monitor of events that occur in the teaching environment's database (Claroline). The monitor performs actions in the occurrence of a critical event. Figure 2 represents the Claroline structure integrated with the event's monitoring system.

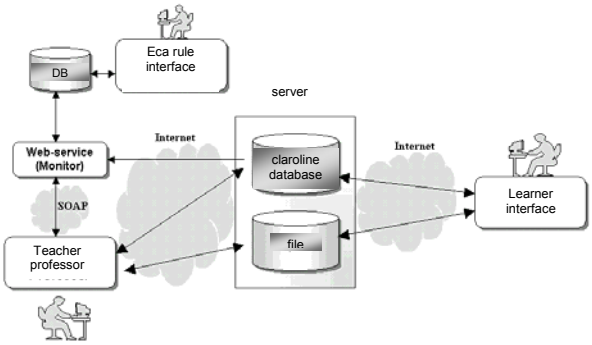


Figure 2 - Claroline integrated to the module of events monitoring

The monitor consists basically of an interpreter of SQL commands. It receives as parameters: an SQL command from Claroline, the user's ID and the course ID. With these parameters it is possible to identify the event (*INSERT, DELETE, UPDATE*) and the table where it is taking place. The monitor of events was implemented as a web service, offering a practical and efficient way for applications communication and data exchange over the Internet [7].

The communication between Claroline and the monitoring web service is carried out through SOAP (Simple Object Access Protocol); a protocol based on XML (Extensible Markup Language) and establishes a communication path among applications in different domains [7]. In the present study, SOAP uses an HTTP transfer protocol to send messages of remote functions request. HTTP facilitates the transportation of SOAP messages among systems because *firewalls* usually do not block the access to the HTTP port. The fact that XML messages are text-based justifies the possibility of communication among different platforms.

The Claroline system underwent small changes in order to call the monitoring web service. We highlight that the performance of the SQL command by Claroline remains the same. The monitoring web service only determines the action to be taken in case there is an ECA rule defined.

4.2 Implementation tests

In order to validate the alarm system integration with the Claroline environment, some alarms were defined and activated in the environment. Two types of actions were implemented: sending e-mails and calling of remote functions through a web service.

The first action consists basically of sending e-mail to all students of the same class about the scheduling of a new task. In Claroline, scheduling works as follows: the teacher accesses the system and schedules a test or homework. Scheduling data are stored in the Claroline database and can be seen when the student accesses the course's page. To make the teacher's work easier and help the student, the monitor of events identifies the addition of a new task in the scheduling table and sends e-mail to all students with the new item. This happens in a transparent mode to the teacher, who does not need to interfere in the process. The administrator registers the message that will be sent.

Besides, an action that calls a remote procedure, a web service, was also implemented. In order to test this functionality we used the model described in [8], which defines a central data repository that contains information about students from different distance teaching environments. It was necessary to develop an action that enabled Claroline to send its student's data to the repository, so a web service was created and named Notify_Data. The function implemented to perform this action is specific for events in the Claroline's table that stores the student's registers. Its main goal is to call the remote procedure named Notify_PersonalData from the *Web-service* defined by Musa. This procedure warns the repository that new data have been inserted in the

Claroline database and requests authorization to insert them in the repository.

During informal conversation with some students and teachers who used the system and received alarms, we noticed they found some alarms interesting and some inconvenient. Alarms that suggest content to be studied or exercises were considered positive. Alarms that send the students an e-mail reminding them to access the course after an idle period were considered inopportune.
The teachers suggested an alarm should be sent after a certain period of time, i.e., the alarms generated should be sent to the user gathered in a batch. This suggestion can be applied to some kinds of alerts.

5 Conclusions and Future work

The alarm system presented in this article was implemented as a generic module, so that it can be attached to other web-based teaching environments. Aiming at validating the use of this system in a e-learning system, it was integrated into the Claroline environment, which is being tested by a great number of users, students and professors, of the Computer Science Institute of UFRGS. An advantage was the choice for an environment developed within the policy of open code software. The entire source-code and the database could be fully accessed and changed whenever required. One of the main difficulties found during the integration was the fact that the alarm system needed the e-learning system to inform data about the teaching environment database. Besides, the alarm conditions are expressed as SQL queries, what constrains the use of the environment. A different way of solving these drawbacks is under investigation. The information of data regardless of the administrator's presence is under study too, as well as a language or editor that has conditions already defined to be used at moment of the alarms edition.

As the alarms are being generated in the Claroline environment, it is possible to capture data that are essential for the validation of this model both for student's activities follow-up and as a way of keeping them motivated in the course they are taking. Some results have already been obtained and were shown to be quite promising, being important indicators of the system's success. Now we are offering the possibility of alarms management for the full set of courses being teatched at II-UFRGS to gather a big amount of real experimental data. We hope this experience can give a step further on the contributions to make Distance Education a feasible alternative in the educational process.

6. References

[1] Viccari, Rosa. (1998) Artificial Intelligence and Educacional Systems. Journal of School n.2, Olivais, vol I, 1998.

[2] Claroline System. Available at: http://www.claroline.net

[3] Joice Lee Otsuka, Heloisa Viera da Rocha: An Agent-Based Approach to Support Formative Assessment. ICCE 2002: 1486-1487

[4] Roda, C. A multi-agent system for advising and monitoring students navigating instructional Web sites." In 6th International Conference on Information Systems Analysis and Synthesis (ISAS 2000*),* Special session on Cooperative and Distance Learning. Orlando, Florida. July 2000. 505-509

[5] Niegemann, H. (2002) EDASEQ – A *log* file analysis program for assessing navigation processes – Proceedings of the 8th International Conference on Computers in Education / International Conference on Computer-Assisted Instruction 2000. Taipei, Taiwan.

[6] Peled, A. (1999) Logging for Sucess - Advancing the Use to Improve Computer Mediated Distance Learning - Journal of Educational Computing Research. v. 21, n. 3, 1999.

[7] Tsalgatidou, A. and Pilioura, T., An Overview of Standards and Related Technology in Web Service. In Distributed and Parallel Databases, vol. 12, p. 135 – 162, ISSN:0926-8782, 2002.

[8] Musa, D.; Palazzo, J. M.O.; Sharing Learner Information through a Web services based Learning Architecture. IN WEB INFORMATION SYSTEMS MODELING, WISM, 2004, Riga, Latvia. CAiSE Workshops , Riga, 2004. p.122–131.

Modeling Interactions between Web Applications and Third Party Systems

Nathalie Moreno and Antonio Vallecillo
Dpto. de Lenguajes y Ciencias de la Computación
Universidad de Málaga, Spain
{vergara,av}@lcc.uma.es

Abstract

Web-based applications are no longer isolated systems. Now they need to interoperate with external service providers and legacy systems, which are available in a wide range of different platforms, and may follow disparate communication mechanisms. Modeling the interactions between these systems is not simple, and needs to be properly addressed within any model-driven development scenario. Many of the existing Web Engineering proposals do not take this fact into account, or else they address it in a very simplistic way. In this work we use an MDA approach for encapsulating the different interaction abstractions and mechanisms into a separate platform-independent level, and show the transformations required to produce platform-specific models depending on the particular details and interaction mechanisms of each technology platform and middleware.

1 Introduction

As the demand and the number of available distributed Web applications grows, so does the need to easily design, deploy, maintain, *integrate* and *interconnect* such Web applications in heterogeneous environments. MDA [17, 5] seems to be one of the most promising approaches for addressing these issues: it provides the right kinds of abstractions and mechanisms for improving the way applications are integrated and interconnected nowadays.

A proper integration approach requires a structured and efficient way to assist software architects and developers achieve such integration not only at implementation level, but also during all phases of the development process. In this regard, any integration with legacy data and external services at the PIM level requires modeling them, too (their structural features, behavioral descriptions, etc.)—allowing the manipulation of the external entities of such systems like native elements of our models. Special care should be taken in this case with the bridges that connect the system with its external partners applications, for which transformations are also needed, as mentioned in [10].

Although a priori there are no major problems with this approach, we may face different kinds of incompatibility issues when trying to integrate external pieces into the system (e.g., external services or legacy applications). For instance, the interface of the services required by our application (as specified in one of the PIMs) may not match the interface of the actual service, as provided by the external service provider. There is no problem if these incompatibilities are explicit because they can be easily detected and corrected—as happens with signature incompatibilities, for example. These situations can be treated with the use of adaptors, wrappers, or any kind of adaptation technique.

The major problem appears in the cases of implicit assumptions on the interaction models and mechanisms followed by clients and servers. Normally, these assumptions are implicitly made by software developers with previous knowledge about how the target platform(s) work. Whenever all of the application is generated from the initial PIMs using a single platform technology, and therefore all parts follow the same interaction models and patterns, this problem does not arise. However, when we need to work with external entities, the interactions models of each party should be made explicit so as to be able to detect and resolve potential inconsistencies and conflicts at design level.

This work presents an approach for modeling the communication mechanisms between a Web application and its related external systems. It makes explicit both the programming abstractions through which the client and service provider perceive and use the communication, and certain implementation choices about the selected target platform that are generally implicit. This is specially relevant in those contexts in which several platform technologies may be simultaneously used.

In general, there is no standard way of describing implementation decisions such as concurrence, security or transaction, in order to get computationally complete PIMS, i.e., PIMs that contain all the information required to produce real program code. Several approaches address this issue

by identifying these concerns at different levels of abstraction. For instance, Almeida et al. [1, 2] introduce the *abstract platform* concept, which defines the characteristics required for the mappings onto the set of concrete target platforms, which are considered in an MDA design process. Following a different approach (that uses a UML profile) Witthawaskul and Johnson [21] define the *unit of work* concept which can be applied to a UML operation to support platform independent transaction modeling. Similarly, our work follows an approach based on marks (using a UML metamodel) that guide both the PIM to PSM transformation, and also the PSM to the Implementation Model transformation. They represent interaction model capabilities and services provided by potential target platforms abstracted away and specified in a platform-independent way.

Another controversial issue is to do with the place where implementation decisions are expected to be specified: (i) directly in the PIM; (ii) in the target platform model, or; (iii) in the transformation model. The MDA community still struggles to deal with this issue, as a quick look at the discussions happening in the MDA mailing lists clearly reveals.

The structure of this document is as follows. After this introduction, Section 2 provides a brief description of the interaction styles supported by technologies like CORBA, Enterprise Java Beans, J2EE, Web Services or .NET. After that, Section 3 derives a UML metamodel based on existing similarities found among the previous interaction models. Using this metamodel, Sections 4, 5 and 6 show how to apply it in a service-oriented scenario. Finally, Section 7 draws some conclusions and outlines some future research activities.

2 Interaction Models for Web Applications

Currently, Web applications need to interoperate with third party systems (external portlets, Web services or legacy applications) in a variety of ways—interaction models—which reflect the heterogeneity of applications built upon disparate implementation technologies such as J2EE, CORBA or .NET. Generally, each middleware technology has its own interaction model, although traditional client-server interaction patterns are likely to be common.

A Web application may be required to communicate with a great variety of systems in different address spaces and running on heterogeneous platforms—which use different communication abstractions and interaction models. Here we will briefly describe the interaction styles of the most commonly used technologies, which are able to connect applications implemented using heterogeneous technologies.

Three main technologies support nowadays communication between modules of disparate systems, hiding platform and language specific details: CORBA, Enterprise Java Beans, J2EE and Web Services and .NET.

CORBA Service Provider. To request a CORBA service provider, the client may follow one of two approaches [8, 15, 19]: (i) a static invocation method or (ii) a dynamic invocation method. In the former, the client has to acquire an object reference to the CORBA object at compile-time. This reference is used to initiate a proxy object that represents the remote object in the client's address space. For generating the proxy implementation, an IDL specification of the CORBA object is required and compiled into the client program. IDL specifications can define both synchronous (request/reply) operations and asynchronous (one-way) messages.

For dynamic invocations there is no information about the types and interface specifications of the required CORBA service. The client can look this information up by querying an Interface Repository (a service that provides IDL definitions at run-time). In consequence, a client request consists of operations for setting the name and parameters of the request and retrieving the returned values or an exception at run-time. Once the client has acquired a valid remote object reference to the CORBA server object, it can call the server object's methods as if the server object resided in the client's address space. The mapping of the object name to its implementation is handled by the Implementation Repository.

Enterprise Java Beans/J2EE Provider. For a client to call a business method, it needs to go via an EJB object (a generated Java class based on the Component Interface). This means that a client never accesses an enterprise bean directly [4, 20]. Firstly, the client has to call a factory object (which is the EJB Home Object) to either locate an existing EJB object or create a new one EJB object. Once generated during compile or deployment time, EJB objects act as bridges between the client and the bean instances [18].

There are several types of EJBs: *session beans, entity beans and message-driven beans*. The two former kinds of beans provide their interfaces to allow remote clients to invoke them. However, message-driven beans do no make their interfaces public. On the contrary, a message-driven bean listens for messages that are sent using *Java Message Service* and processes them anonymously (asynchronous invocations).

An XML file describes how an Enterprise Java Bean should be assembled and deployed, its name, and other external dependencies of the bean.

Java/RMI. This mechanism is tied to the Java programming language and virtual machines [9]. RMI allows operations on Java objects to be invoked. The client should first contact an RMI registry, and request the name of the

service. RMI URLs identify services, including the hostname on which the service is located, and the logical name of the service. Then, the registry will point the client in the direction of the service it wants to call. The mapping of Object Name to its Implementation is handled by the RMI Registry.

RMI generates proxies and stubs from Java interface definitions. Furthermore, RMI uses Java's capabilities for dynamic linking to load the classes of parameters or returned objects over the network, allowing clients or servers to receive objects of classes unknown at compile time.

Web Services & .NET Provider. In order for a client to be able to employ a Web service, the client should know where the Web service resides and how to invoke its methods (that is, how to serialize the call to the Web service and how to deserialize received messages from the Web service). This information is provided by the WSDL specification of the Web service [22]—an XML document that specifies the data types of the messages, the protocols that are accepted, the Web service's endpoint, and the bindings.

Since the notions involved in creating the SOAP message to be sent to the Web service, making the actual HTTP request, deserializing the HTTP response, etc. could be complex, they are abstracted using a proxy class. Such a class encapsulates the complexity of calling a Web service and reveals a simplified interface [11]. From the client application's perspective, the Web service is simply a local component—the client doesn't have to worry about the specifics of how to serialize a SOAP message, or how to make a HTTP request.

3 Modeling interactions

3.1 Basic Interactions

The basic interaction model works according to the three-step process shown in Figure 1, being different interaction models supported by combinations of this configuration—mainly combinations of the second and third steps.

Step 1. A service provider publishes a description of their services in a publicly accessible registry.

Step 2. A service requestor discovers those services by querying the registry and binds to the selected service. (Note that we will call the service requester a *client*)

Step 3. The client interacts with service provider.

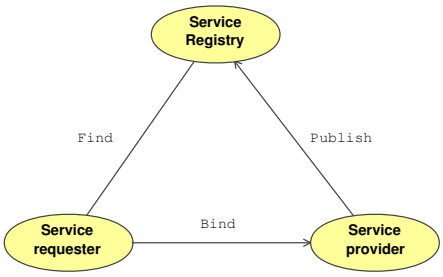


Figure 1. Basic Interaction Model

According to Figure 1, an interaction between two endpoints can be defined in terms of:

- The set of messages accepted by the service provider (*Provided Interface*)
- The set of messages required by the client (*Required Interface*)
- A protocol that defines the partial order between the exchanged messages.
- The programming abstractions through which the client and server view the protocol (*the client-side and server-side programming interfaces*). This is important, because these programming abstractions encapsulate the agreement between both parties on the data format, the mechanism for transforming and reconstructing the object state into this format, the transport protocol, etc.

Most approaches just focus on the first three points. However, the fourth one is not explicitly stated or modeled anywhere; instead, it is usually implicitly assumed by both the client and the server, and therefore hard-wired into their models, transformation rules, and code. This is not flexible and does not provide the platform-independence required in a true MDA approach. Besides, these assumptions are usually made separately, which may cause other contradictory choices. Thus, being able to express this kind of information—particulary the last point—in a Platform-Independent way is a step forward to achieving abstracts models established in more detail that allow code-generation MDA tools to obtain real implementations.

3.2 Identifying Model Elements and their Relationships

In our proposal we have tried to use existing UML elements as much as possible, in particular UML 2.0 elements because they provide some useful architectural concepts

UML Base Element	Stereotype
Port	≪ServerPort≫
Port	≪ClientPort≫
Port	≪StubClient≫
Port	≪ProxyClient≫
Port	≪DynamicClient≫
Interface	≪InterfaceSignature≫
Interface	≪ProvidedInterface≫
Interface	≪RequiredInterface≫
Assembly Adaptor	≪Interaction≫

Table 1. Summary of the stereotypes used

and mechanisms for our purposes. Table 1 shows a summary of the profile we have defined for representing these concepts. In particular, we consider each system as a UML 2.0 *Component*, which represents "a modular part of a system that encapsulates its contents, designs as well as implementations features, without losing the ability to describe deployment information and being replaceable within its environment" [16].

Component interactions are carried out through a layer of abstraction that allows clients to instantiate and access the methods of the external services provider. In this sense, we can define one or more *Ports* through which a *component* invokes and receives method calls.

Since each endpoint can act as either a *provider* or a *client* in each of the Web interactions in which it plays a role, we have modeled causality by a *Port* stereotyped as *ClientPort* or *ServerPort*.

The interaction between a *ServerPort* and a *ClientPort* falls into one of the following categories:

- Synchronous invocation. The *ClientPort* invokes a remote procedure and blocks it until a response or an exception is received from the *ServerPort*.
- Asynchronous invocation. The *ClientPort* invokes a remote procedure and continues processing without waiting for a return, although the returned value will be received at any moment.
- One-way invocation. The *ClientPort* invokes a remote procedure but does not block or wait to receive a return since it will not receive a return value.

Initially we will consider that each *Port* is associated with only one *Interface*. More precisely, a *ClientPort* is associated with a *RequiredInterface* and a *ServerPort* is associated with a *ProvidedInterface*. A *ProvidedInterface* specifies public operations that are remotely available. On the other hand, *RequiredInterfaces* complement *ProvidedInterfaces* and describe those external features on which a system depends to implement its functionality.

Ports describe how a *System* interacts with its environment. They are different to *Interfaces* because *Interfaces* contain just syntactic information about the methods provided by a *System*, while *Ports* encapsulate the required business logic that allows a *Requirer* to interact with a *Provider*, tying that business logic with a concrete "implementation choice". Note that in many cases some implementation choices will only be supported by certain target platforms.

Each *Port* has an associated a *Protocol* that defines the partial order in which the post owner object expects its methods to be called, and the order in which it invokes another object's methods. *Port's Protocols* show a global perspective over its constituent external applications protocol descriptions. For simplicity we have supposed that each *Port* is associated with only one *Interface*, and hence *Port's protocols* will coincide with *Interface's protocols*.

ServerPort's Protocol can be describe as text files using BPEL4WS or WS-CDL specifications, for example. *ClientPorts* interfaces can be augmented with behavioral descriptions based on *protocol state machines* that define usage constraints among features of the associated interface. Many aspects of the *ClientPort* are determined by the external system which the client connects to. In consequence, the *ClientPort* can be classified into three main categories based on the third party system that they can interact with:

(*i*) *Stub Clients* are never required to be downloaded or distributed to clients and they are specific to a certain protocol, transport option and server requirer (accorded at compile time). The client must obtain a reference to the *Stub* before using it, which represents an instance of the server provider. In order to obtain it, both the remote interface and its implementation have to be available so the client relies on an implementation-specific class.

(*ii*) *Proxy Clients*, as *Stub Clients*, refer to static invocation of server provider methods. They are not portable across implementations either—in this case, the code for the *Proxy Client* is created during runtime, but the reference to the interface specification of the external provider is obtained at compile-time.

(*iii*) *Dynamic Clients* can access a service discovering its interface description dynamically. In the same way, they can invoke server provider methods at runtime. This implies an extra work at runtime to fetch and process the server interface.

At this point, a benefit of using *Ports* is that the constraints and requirements on the communications between applications can be modeled without forcing software developers to take into account the platform specific notions in their designs. In this way, the designs can be reused to

be run on different platforms (hence following the platform-independence philosophy dictated by MDA).

The kind of *Client Port* to be used is important, and strongly influences the kinds of client-side artifacts that need to be generated at development-time. Both *Stub* and *proxy* clients require the complete interface specification of the external services. That is, the client does not need to discover the required service but instead it has, at development-time, to know the external system's details (location, configuration file, WSDL or IDL URL, namespace, etc.). In contrast, *Dynamic Clients* must dynamically discover and invoke an external system without any prior knowledge of its details (signature of the remote procedure or the name of the service). For a *Dynamic Client*, there is no coupling between the service interface and the client. This makes the client code easy to modify if the external systems specifications change.

On the other hand, one of the most significant differences between *Stub Clients* and *Proxy Clients* is how external functionality is invoked. For the former, the client-side programming interface is embedded inside the client business logic. On the contrary, for *Proxy Client* and *Dynamic Client*, the client-side code is packaged apart from the client application.

Please note that the selection of these *Ports* only affects the client side. From the server perspective, it only receives and returns messages which are identical for all client types.

3.3 Adaptors

In case there is a strong requirement of using an external service provider (e.g., for *Stub Clients*), the software designer can specify what should be done if the behavior/specification of both parties is incompatible. Since we have the interface of the required external systems (*provided interfaces*) available, we can carry out static checking for comparing them and determining whether they fulfil our requirements (*required interfaces*).

If not, the designer can decide at designing-time to implement an intermediate business logic (*adaptors*) that conforms to a given interface or employs a required external system. *Adaptors* mediates between the *ClientPort* and *ServerPort* interactions, resolving *service provider* and *service requester* differences at interface and protocol levels.

4 Example: The Travel Agency

In order to illustrate the use of interaction patterns in the definition of Platform Independent and Platform Specific Models, let us consider a Travel Agency service that sells vacation packages to its customers. The packages include flights, hotel rooms, car rentals, and combinations of these. External service providers include transportation companies (airlines, hotels and car rentals) and financial organizations (credit companies and banks).

To book a vacation package, the customer will provide details about his preferred dates, destinations, and accommodation options to the Travel Agency System (TAS). Based on this information, the TAS will request its service providers for offers that fulfill the user's requirements, and will then present the list of offers to the customer. At this point, the customer may either select one of the offered packages, reject them all and quit, or refine his requirements and start the process again. If the customer selects one of the packages, the TAS will book the individual services to the corresponding transportation companies, and charge the customer.

The straightforward application of MDA to develop a system is based on the following steps:

Step 1 Create class diagram (PIM) describing object model.

Step 2 Mark PIM elements with stereotypes.

Step 3 Customize the marked PIM with annotations.

Step 4 Specify the target platform.

Step 5 Generate a PSM.

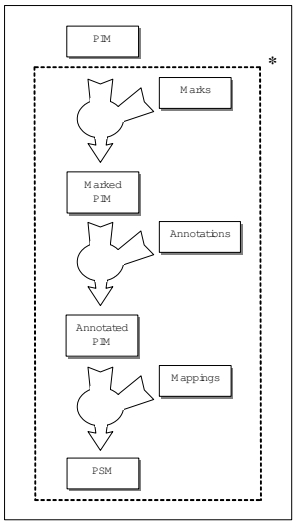


Figure 2. The PIM to PSM iterative process

In general, the MDA software development process becomes an iterative model transformation process where each step transforms one (or more) PIM of the system at one level into one (or more) PSM at the next level until a final implementation model is reached (see Figure 2). Here,

an implementation model is just another PSM, which provides all the information needed to construct a system and to put it into operation).

Note that we will call the last platform *technology platform* (i.e., the one that provides the executable PSM, or *implementation*). The intermediate platforms that transform PIMs into PSMs that will be used as PIMs in the next step are considered as *abstract platforms*.

Given that an element of the PIM may be marked several times with marks that come from different metamodels, it will be transformed according to each of the mappings. The semantic of the resulting marked element is given by the gathered features through the MDA model transformation process.

In our approach we need to go though two main phases. Firstly, we need to identify the system scope and boundaries, i.e, which services will be provided by our system, and which ones will be externally required. The result of this phase is a high-level architectural view of the services and components of our global system. In the second phase, we need to determine the concrete platforms and communication mechanisms between our application and the external systems identified previously.

5 Identifying the scope and boundaries of our system

In our previous work [14], we presented a model-based framework that allows the high-level integration of Web applications with third party systems aligned with the MDA principles. It enables the manipulation of the external entities and systems as native elements of our models.

At design level, software developers are able to specify/mark: the system elements that require code generation; the system elements that will be remotely accessed using its provided interface specifications and implementations; the system elements that need to interact with others; and the system properties that are used for identifying them. All this is done in this first phase in a platform-independent manner, i.e., independently from the communication asbtractions and mechanisms used, and the platforms in which our system and the external services are implemented. These details will be added in the second phase.

In the first place we need to create the PIM of the system, which in our case is shown in Figure 3. It focuses just on the operations of the system, while abstracting away the rest of the details (software architecture, distribution, system boundaries, communication protocols, implementation platforms, etc.). This solution is specified in terms of UML packages and their interconnections in a platform independent manner, where no implementation decisions have been explicitly stated (this greatly simplifies the PIM of the application making it reusable across different target platform environments).

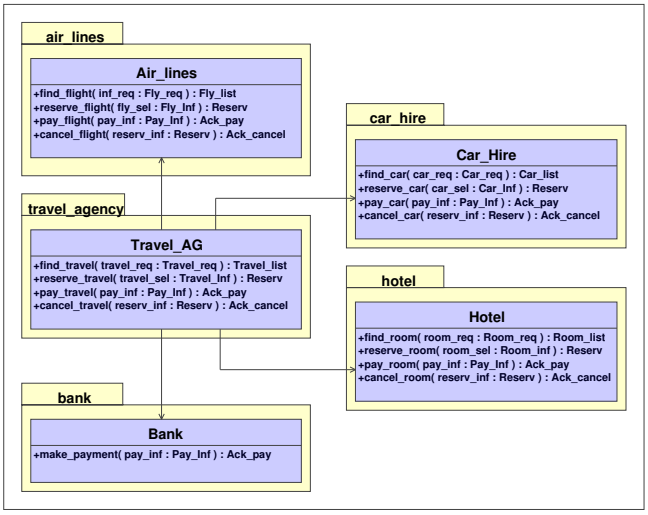


Figure 3. The TAS PIM

As previously mentioned, any integration to legacy data and services may require that the interfaces to those elements are also modeled. The kind of information that is available from them will allow us to check whether they match our requirements or not, as described by the system model [13]. More precisely, this information should be able to allow us to:

(*a*) model the component or legacy system (e.g., by describing its structure, behavior, and choreography);

(*b*) check whether it matches the system requirements (this is also known as the *gap analysis* problem [7]);

(*c*) evaluate the changes and adaptation effort required to make it match the system requirements (i.e., evaluate the *distance* between the models of the "required" and the "actual" services, see e.g., [12]); and

(*d*) ideally, provide the specification of an adaptor that resolves these possible mismatches and differences (see e.g., [6]).

The integration of third party systems with a Web application should be addressed at three levels of abstractions (namely, *presentation*, *business process* and *data level*) [14]. For the sake of simplicity, in this paper we will only consider the business process level.

Once the high-level PIM is described, we need to identify the system scope and boundaries, and then build a model of the system with this information. That target model (PSM) will be built by transforming the original PIM using marks. To identify the elements in

the TAS PIM that should be transformed in a particular way, we will use the stereotypes ≪ExternalSystem≫ and ≪ExternalAssociation≫. An ≪ExternalSystem≫ defines any other external system interacting with the system under consideration. In the same way, an ≪ExternalAssociation≫ defines an interaction between the system under deployment and an ≪ExternalSystem≫.

Implicitly, each type of model element in the PIM is only suitable for certain marks, which indicate what type of model element will be generated in the PSM.

Marks are not a part of the platform independent model although they appear on the marked PIM (see Figure 4).

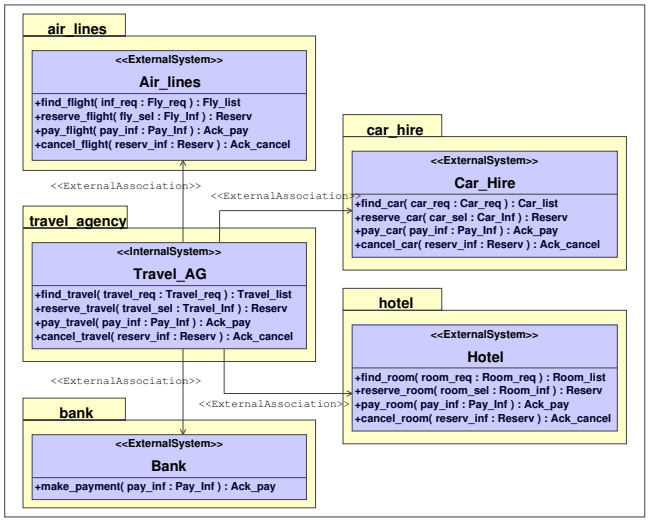


Figure 4. The marked TAS PIM

Note that the marked PIM is, by definition technology independent. In consequence, the prefix "External" used by the stereotypes ≪ExternalSystem≫ and ≪ExternalAssociation≫ in Figure 4 does not imply any implementation decisions. Instead, it is only used to limit the system scope that has to be development.

Once we have the marked PIM, we need to transform it into a PSM that can be translated into a target implementation code. As "platform" we will use here the UML 2.0 constructs and infrastructure for describing software architectures, because what we want to build in this phase is the software architectural description (i.e., model) of the system. This transformation will be guided by the following mapping rules:

- **Packages transformation.** Each UML package is mapped to a UML ≪Component≫ initialized with the same as its corresponding UML package.
- **Classes transformation.** The UML class stereotyped as ≪InternalSystem≫ or ≪ExternalSystem≫ is mapped to a UML ≪Class≫ holding the same characteristics as its original (name, attributes and operations).
- **Associations transformation.** For each UML association stereotyped as ≪ExternalAssociation≫ two component ports will be generated, each one as Association ends of that relationship. Ports will be associated to the UML ≪Component≫ derived in the previous step. Its behavior is defined in terms of an interface associated with that port, which specifies the nature of the interactions that may occur over that port. Thus, the port interface's name is given the value of the UML class name from which it derives and its operations correspond to its UML class operations.
- **Transformation of Association's ends.** For the endpoint of an ≪ExternalAssociation≫ stereotyped as ≪InternalSystem≫, a usage dependency from the port to the interface is generated, showing how the ≪InternalSystem≫ provides a set of services.

  For the endpoint of an ≪ExternalAssociation≫ stereotyped as ≪ExternalSystem≫, an implementation dependency from the port to the interface is generated, showing the services required by the ≪ExternalSystem≫.
- Finally, assembly connectors are defined from required Interfaces to the corresponding provided Interfaces.

Applying these mapping rules on the PIM in Figure 4, the PSM shown in Figure 5 is obtained.

As previously mentioned, the MDA software development process is an iterative model transformation process whereby a PIM is transformed into a PSM, which in turn becomes the PIM for the next transformation—until a final PSM (the system *implementation*) is reached. What counts as a platform depends on the level of abstraction, and the kind of system being developed.

6 A Platform Specific Interaction-Model

Once we have the (UML 2.0) architectural description of the system, that identifies its scope and interactions with external services, the next phase focuses on the specification of such external interactions using the particular platforms and communication mechanisms of the required services. By adopting an MDA transformation process based on marks and annotations, we have to define the marks and transformations required.

Basically the information that the transformation process has to generate from the marked PIM is: the communication mechanisms between the *Components*; how the communications will be carried out; and the information that

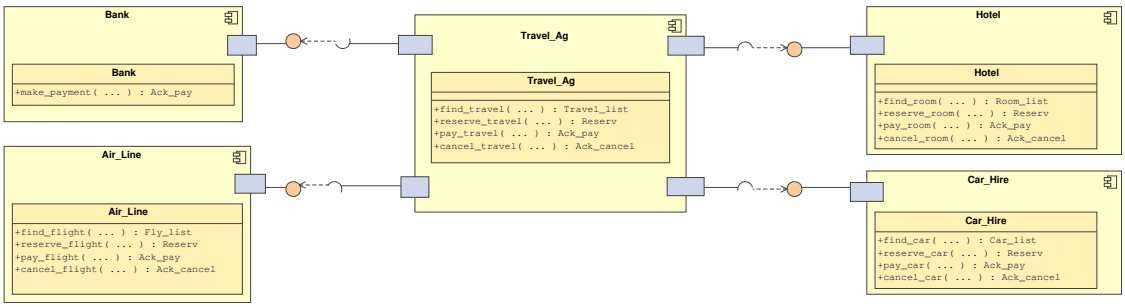


Figure 5. The PSM after applying the MDA transformation

describes the architecture of the Web application. Therefore, the model shown in Figure 5 has to be marked again to specify that information.

Once we have applied previous transformation rules on the PIM, the resulting PSM is also platform-independent. We will mark them with decisions which are considered and taken in the context of a specific implementation design based on the concepts discussed in Section 3:

- Ports that specify services provided by external entities are stereotyped as ≪ServerPorts≫
- Ports that specify required services are stereotyped as ≪ClientPorts≫.
- Finally, assembly adaptors connecting interfaces have been stereotyped as ≪Interactions≫.

The resulting model is shown in Figure 6.

Now it is time to include information about the technologies used to interact with the external services.

In the particular case of the Travel Agency System, we are going to make use of external service providers which include transportation companies (airlines, hotels and car rentals) and financial organizations (credit companies and banks). For illustration purposes we have selected different technologies for each external service. More precisely:

- A CORBA implementation of the Hotel Service. As previously mentioned, in order to participate in an interaction with a CORBA server application, the client (that is, our Travel Agency Service) must be able to get an object reference for a CORBA object and invoke operations on the object. To accomplish this, the client needs information about references to the environmental objects that provide services for the CORBA application we plan to use and the IDL specification for implementing a stub-style invocation. Figure 7 shows how this information is specified using notes associated to its corresponding stereotypes.
- Another CORBA implementation of the CarHire Service. In this case, we plan to implement a dynamic interaction pattern so the IDL file will be looked up in an Interface Repository where it must be stored. In that sense, no IDL file has to be provided by the external server provider.

  The exact steps taken to access the Interface Repository depend on whether the client is seeking information about a specific object, or browsing the Interface Repository to find an interface. In both cases, before a dynamic client can browse the Interface Repository, it needs to obtain the object reference of the Interface Repository to start the search. Once the client has the object reference, it can navigate the Interface Repository, starting at the root.
- The two other external services are supposed to be available as external Web services. Their respective WSDL interface descriptions are required as illustrated in Figure 7. Additionally, the code for the interaction with the Airline Web service relies on an implementation-specific class since it uses an stub-style. This means that its implementation should also be available.

At this point, we also need to decide on the implementation technologies and platforms of our own system. Imagine that we decide to implement the Travel Agency using Java and Web Services technologies.

In this case we could use the transformation rules of any of the existing approaches for converting our marked and annotated PIM (in Fig. 7) to the corresponding PSM (shown in Fig. 8). For instance, we could follow the approach by Bezivin et al. [3], and then proceed according to the following steps:

1. Code each service endpoint interface and its implementation class. A service endpoint interface declares the methods that a remote client may invoke on the service. In this case, each UML class is mapped to

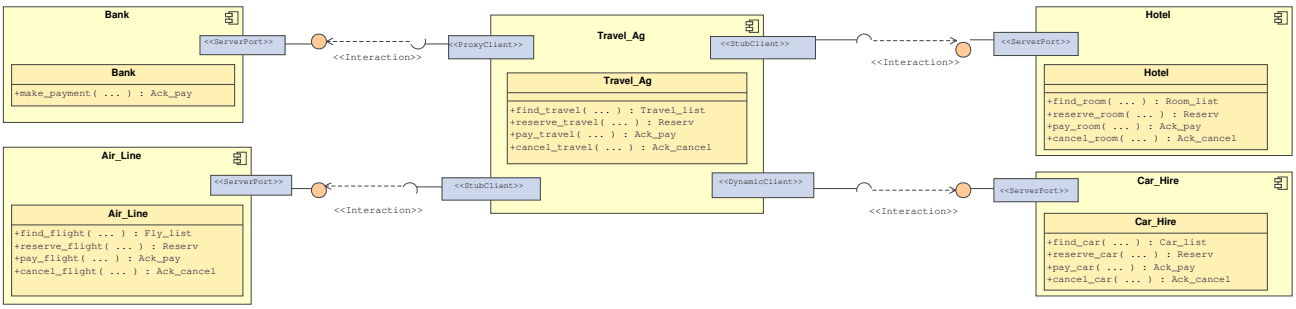


Figure 6. Marked PIM

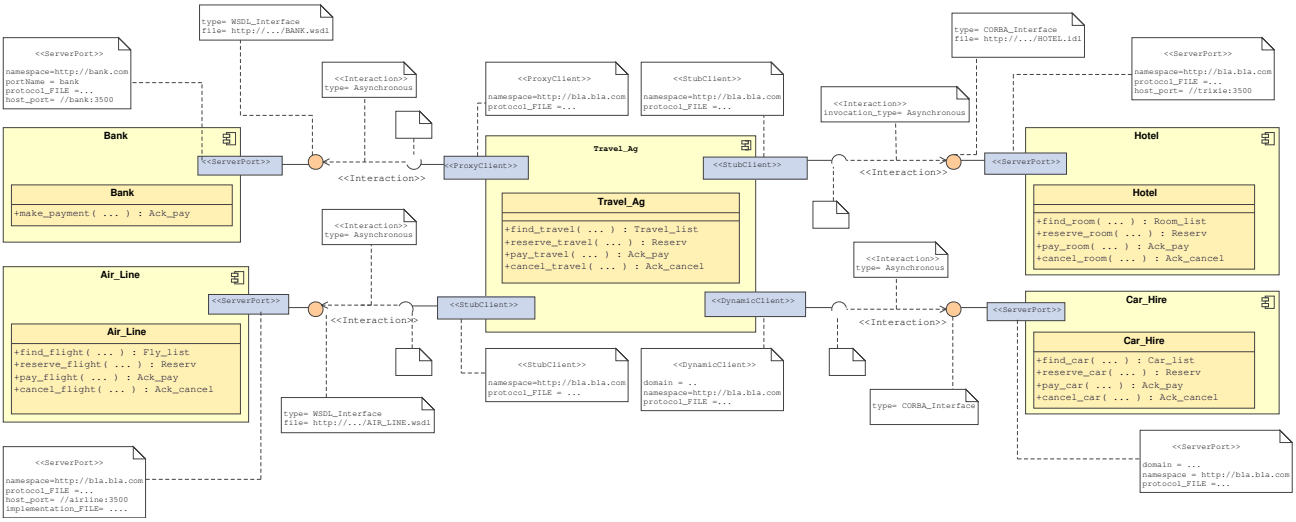


Figure 7. The Annotated TAS PIM

a ≪JavaClass≫ initialized with the same characteristics of its corresponding UML class. Based on this, the ≪JavaInterface≫ is also derived.

2. Build, generate, and package the files required by the service. In this case, each UML class is also mapped to a ≪WSDL Specifications≫: ≪WSDL types≫, ≪WSDL operations≫, ≪WSDL bindings≫ and ≪WSDL services≫.

3. Deploy the service. Four deployment files are required in our case: web.xml, jaxrpc-ri.xml, config-wsdl and config-interface. Thus, the ≪JavaClass≫ is mapped to a ≪JWSDPweb.xml≫, ≪JWSDPjaxrpc-ri.xml≫, ≪JWSDPconfig-wsdl≫ and ≪JWSDPconfig-interface≫ files.

4. Generate client-side abstractions for consuming external services. For the CORBA services, we will add: the client stubs for each interface (interfaceStub.java), the CORBA helper class (interfaceHelper.java) and the CORBA holder class (interfaceHolder.java) that describe everything needed to use the client stub from the Java programming language. For the rest of the services, no more classes are generated (the code is embedded in the ≪JavaClass≫ implementation).

The PSM obtained, shown in Figure 8, includes all the details required to build the final implementation.

7 Conclusions and Future Works

In this paper we have discussed some of the (many) problems that may happen when integrating Web-based applications with external systems. In particular, we have concentrated on the interaction issues due to potential incompatibilities between clients and servers of different implementation platforms and middlewares. Our main contribution is

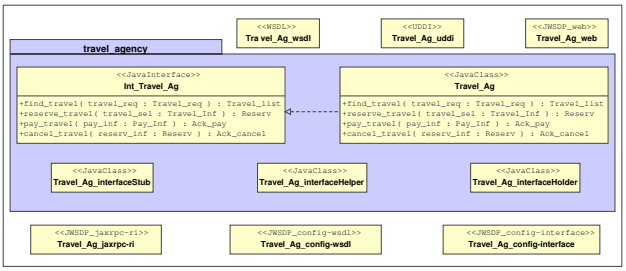


Figure 8. The final TAS PSM

to make such interaction models and mechanisms explicit, so incompatibilities can be detected, and bridges or adapters can be easily built. Besides, we have done it according to the MDA principles, encapsulating those interaction concepts and mechanisms in a platform-independent manner, and then providing transformation rules to the different implementations available of these concepts in the most commonly used platforms and middelwares.

Now that the interaction issues can be solved at this level, we plan to move forward, trying to address two other major issues. Firstly, the (semi-)automatic derivation of adaptors in case of incompatibilities are detected at this level. Secondly, the description of some behavioral and QoS information in the models, in order to deal with these kinds of aspects.

Acknowledgements The authors would like to thank the anonymous referees for their helpful comments and remarks. This work has been supported by Spanish Research Project TIC2002- 04309-C02-02.

References

[1] J. P. Almeida, R. Dijkman, M. van Sinderen, and L. F. Pires. On the notion of abstract platform in mda development. In *The 8th International IEEE EDOC*, pages 253–263, Monterey, California, USA, Sept. 2004. IEEE Computer Society.

[2] J. P. Almeida, R. Dijkman, M. van Sinderen, and L. F. Pires. Platform-independent modeling in MDA: Supporting abstract platforms. In *Proceedings of Model Driven Architecture: Foundations and Applications (MDAFA 2004)*, pages 217–231, June 2004.

[3] J. Bezivin, S. Hammoudi, D. Lopes, and F. Jouault. Applying MDA approach to B2B applications: A road map. *Workshop on Model Driven Development (WMDD 2004) at ECOOP 2004, Oslo, Norway, Springer-Verlag, LNCS*, 3344, 2004.

[4] D. Blevins. Overview of the enterprise JavaBeans component model. In *Component-Based Software Engineering: Putting the Pieces Together*, pages 589–606. Addison-Wesley, 2001.

[5] A. W. Brown. Model driven architecture: Principles and practice. *Software System Model*, 3:314–327, 2004.

[6] R. Bruni, H. Melgratti, and U. Montanari. Theoretical foundations for compensations in flow composition languages. *SIGPLAN Not.*, 40(1):209–220, 2005.

[7] J. Cheesman and J. Daniels. *UML components: a simple process for specifying component-based software*. Addison-Wesley Longman Publishing, Boston, MA, USA, 2000.

[8] M. Henning and S. Vinoski. *Advanced CORBA Programming with C++*. Addison-Wesley, 1999.

[9] JavaRMI. *Remote Method Invocation*. Sun Microsystems, 2004. http://java.sun.com /j2se /1.4.2 /docs /guide /rmi /spec /rmiTOC.html.

[10] A. Kleppe, J. Warmer, and W. Bast. *MDA Explained. The Model Driven Architecture: Practice and Promise*. Addison-Wesley, Apr. 2003.

[11] Microsoft Corporation. *.NET Web Page*, 2004. http://www.microsoft.com/net/.

[12] R. Mili, J. Desharnais, M. Frappier, and A. Mili. Semantic distance between specifications. *Theoretical Comput. Sci.*, 247:257–276, Sept. 2000.

[13] N. Moreno and A. Vallecillo. What to we do with re-use in MDA? *Second European Workshop on Model Driven Architecture (EWMDA-2)*, Sept. 2004. Canterbury, Kent.

[14] N. Moreno and A. Vallecillo. A model-based approach for integrating third party systems with web applications. *Fifth International Conference on Web Engineering (ICWE2005)*, July 2005. Sydney, Australia.

[15] Object Management Group. *CORBA 3.0 - IDL Syntax and Semantics Chapter*, 2002. http://www.omg.org/docs/formal/02-06-39.pdf.

[16] Object Management Group. *UML 2.0 Superstructure Specification*, 2003. http://www.omg.org/cgi-bin/doc?ptc/03-08-02.pdf.

[17] OMG. *Model Driven Architecture. A Technical Perspective*. Object Management Group, Jan. 2001. OMG document ab/2001-01-01.

[18] OpenEnterpriseX. *Just Enough Enterprise Java Beans Concepts*, 2004. http://www.OpenEnterpriseX.org.

[19] J. Siegel. *CORBA 3. Fundamentals and Programming*. John Wiley & Sons. OMG Press, 2000.

[20] Sun Microsystems. *The J2EE 1.4 Tutorial*, June 2004. http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html.

[21] W. Witthawaskul and R. Johnson. Transaction support using unit of work modeling in the context of MDA. Available at http://weerasak.com/Unitf, Mar. 2005.

[22] *Web Services Description Language (WSDL) 1.1*, 2001. http://www.w3.org/TR/wsdl.

Model-Driven Approach for Semantic Web based-hypermedia Applications

Laura Montells, Susana Montero, Paloma Díaz, Ignacio Aedo
Laboratorio DEI. Dpto. de Informática
Universidad Carlos III de Madrid
Avda. de la Universidad 30. 28911 Leganés, Spain
{lmontell@inf, smontero@inf, pdp@inf, aedo@ia}.uc3m.es

Abstract

The existence of web pages that are described semantically via ontologies and metadata conforming to these ontologies is crucial to bring the Semantic Web to life. In this paper we address the problem of developing semantic web-based hypermedia applications through web-hypermedia design methods. For that, we propose a general approach based on extracting an ontology-based design from a traditional model-driven design , thus allowing hypermedia designers to obtain both domain ontologies and annotated documents in parallel with the application design without requiring extra tasks and expert know-how. This approach is presented through a specific hypermedia design method called Ariadne Development Method (ADM) and its software tool, AriadneTool, that automates a semantic extraction process to provide annotated documents in a format suitable for the semantic web, as RDFS and RDF. Moreover, a semantic web platform has been developed in order to enable the publication of the resulting semantic web-based hypermedia application.

1. Introduction

The Web has turned into a medium for sharing knowledge among people, therefore the major emphasis has been placed on how to present it for human readers. However, the increasing amount of information is leading to an information overload. In order to deal with this continuous Web growth, programs must be able to share and process web resources. This is the aim of the Semantic Web [1]: to attain a web of data that can be both human-readable and machine-processable, thus enabling intelligent access to information.

Representing multimedia content (e.g. voice, video, image, and data) with semantics provided by relevant ontology (or ontologies) has been identified as a key challenge for the semantic web. Annotation systems produce semantically tagged pages using web-based knowledge representation languages, such as RDF[1] (Resource Description Framework) and OWL[2](Ontology Web Language). These systems [3] require documents in HTML format and specific domain ontologies in order to produce annotation in a manual or (semi-)automatic way. However, the building of these ontologies is a difficult task that requires extensive knowledge (both a knowledge on engineering and a domain expert) and, in most cases, the result could be incomplete or inaccurate. Moreover, the annotation of documents is usually made over existing static pages as an additional task that takes a long time and human effort, and this process may be incomplete or incorrect if the creator is not skilled enough. Therefore, the success of the semantic web depends on the easy creation both of domain ontologies and ontology-based metadata by semantic annotation.

Although this kind of system is still necessary to convert existing web applications to semantic web applications, annotation would be best performed while designing the web application, not after it is implemented. In this way, we can take advantage of implicit and explicit semantic assumptions made during the conceptual modeling of web-based hypermedia applications to directly generate semantic web applications without any additional tasks and expert know-how.

Combining the use of design methods with ontologies according to the Semantic Web provides us with benefits coming from both of these approaches:

- Makes possible to re-use information designs.
- Offers new uses for existing data.
- Allows information sharing between applications.
- Increases the flexibility of systems that will adapt as requirements evolve.
- Reduces the cost and risk of the application design.

[1] http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/
[2] http://www.w3.org/2004/OWL/
[3] http://annotation.semanticweb.org/tools/

- Makes searching content easier using semantic queries on web application with a great amount of information.

Consequently, we propose the coupling of ontologies into the development process of hypermedia and web applications into the conceptual modeling of the existing web design method [12]. This combination allows us to provide web pages with semantic contents (the annotation process) as well as contextual information about the domain knowledge involved (the ontology domain) upon which the application is being designed. Here we present this approach applied to a specific hypermedia design method called Ariadne Development Method (ADM) [3]. We have developed a software framework integrated by a process which transforms hypermedia application modeling into a global, syntactically and semantically interoperable knowledge base in RDF(S) format using a software support tool for ADM, called AriadneTool [13], and a semantic web platform devoted to its visualization. It is important to remark that AriadneTool is not intended to be an ontology editor, but it provides a way to semantically annotate application contents.

The paper is structured as follows. Section 2 describes briefly how hypermedia design methods can integrate semantic content in a natural way. Section 3 describes the Ariadne Development Method phases and AriadneTool. Section 4 presents the framework where we will integrate the semantic annotation functionality and explains how an application domain ontology and a presentation ontology together with metadata conforming these ontologies are extracted from the different products of Ariadne Design Method ready for publication on a particular semantic web platform. Finally, sections from 5 to 7 presents related works, some conclusions including future work and acknowledgements.

2. How to add semantics to Hypermedia Design Methods

In order to allow hypermedia design methods to include metadata about web resources that are specified during the development process, we propose to integrate ontologies in the conceptual modeling phase. A model is an explicit specification of a set of concepts and relationships between them that defines a description language for a specific domain of interest. Just like models, an ontology includes definitions of basic concepts in the domain and the relationships among them but with a different starting point, as stated in [6]. While the former usually has as foundation to get a successful use in an application development, in the latter an epistemological level underlies to express the intended meaning about what is being conceptualized. So ontologies are the tool that may yield a more concise semantic to design models.

In this approach, the idea is to map the concepts and relations of models used in hypermedia design methods to an ontology language. While the former contribute with their graphical support, the latter adds semantic support. All hypermedia design methods such as WebML [2] or RMM [7] are based on formal models to capture the essence of hypermedia applications. Most formal models can be expressed in terms of ontologies languages; in our case ADM is based on the Labyrinth model [4] to explicitly describe the elements which define the structure and behavior of the application and it can be expressed by means of an ontology web language such as DAML+OIL [11] or OWL as described in [12].

From this coupling, hypermedia design methods and the semantic web can mutually benefit. On the one hand, methods can integrate web standards for expressing metadata about web resources and include formal semantics for checking completeness, consistency and correctness of the design with the respect to the method semantics, thus improving the user's understanding of its use, as in [13]. On the other hand, the semantic web can take advantage of the experience gained from years of research in the hypermedia engineering field through its design methods devoted to obtain well-organized application in aspects of information, navigation, presentation, interaction, personalization and even access control.

In the next section, we present a specific hypermedia design method, the Ariadne Development Method and its implementation, in order to introduce how the underlying semantics of a hypermedia application can be extracted during its design process to produce metadata about information and its presentation applying the approach presented here.

3. The Ariadne Development Method and the AriadneTool toolkit

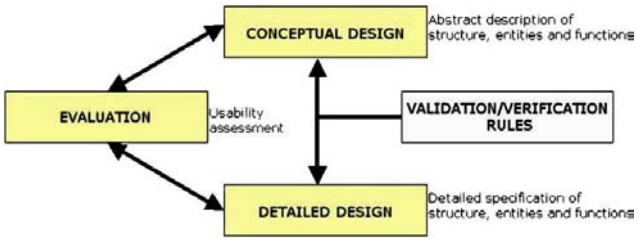


Figure 1. The ADM process

In a nutshell, the Ariadne Development Method establishes a systematic process composed of three phases as illustrated in Figure 1. The *Conceptual Design* is focused on identifying abstract types of components, relationships and functions; the *Detailed Design* is concerned with specifying the system features, processes and behaviors in a detailed

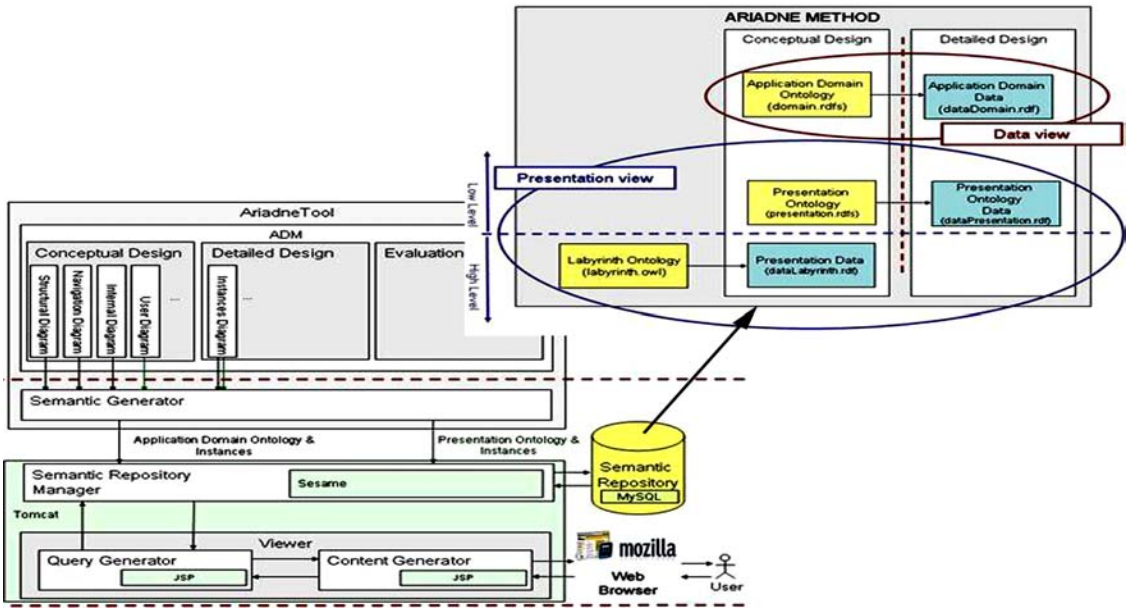


Figure 2. Overview of arquitecture

way in which the application might be generated; and, finally, the *Evaluation* is concerned with the use of prototypes and specifications to assess the system usability. Furthermore, each one of them proposes a number of design products to specify and produce hypermedia and web applications. The arrows shown in Figure 1 mean that the method does not impose any kind of sequence among phases, letting developers decide the best way to face their work according to their needs. Moreover, the method provides a number of *Validation and Integrity Rules*, both at the intra and inter phase level to check completeness, consistency and integrity among the various design products. AriadneTool [13] is an environment designed to develop hypermedia applications based on ADM supporting fast-prototyping in HTML, XML, SMIL and RDF, as well as automatic generation of documentation about the design process.

We will next explain how AriadneTool extracts knowledge about the application domain and expresses it in RDF/S format, suitable for the semantic web.

4. Automatic extraction of semantic information from the design process

Before explaining the extraction process, we will describe the architecture of our approach which is depicted in Figure 2. It is made up three component modules. The **Semantic Generator** recollects semantics and presentation characteristics associated to a hypermedia application designed with AriadneTool. It is implemented as a Java module within the tool and uses Jena[4] for creating, modifying and inferring knowledge about the modeling, and expresses it in RDFS and RDF format. The **Semantic Repository Manager** and the **Viewer Module** are external applications that are needed to store and manage semantic information in order to be presented later to the user. The Semantic Repository Manager uses the Sesame[5] repository for managing the semantics from the application. It uses RDQL[6] as query language and MySQL to store the metadata generated by the Semantic Generator. The Viewer Module is implemented with JSP[7].

As shown in Figure 2 the semantics extracted are stored on the semantic repository according to two different points of view: the *data view* and the *presentation view*. The following subsections briefly describe how AriadneTool extracts semantics from the different products of ADM through a simple example about a research group website, which provides information about its members, research areas and publications. Moreover, some pieces from the annotation produced in RDF and RDFS format are included.

4.1 The data view

AriadneTool extracts the ontology and its instances about the data of the research group website example from the Conceptual Design and the Detailed Design, respectively.

[4] http://jena.sourceforge.net
[5] http://openrdf.org
[6] http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/
[7] java.sun.com/products/jsp

This process generates an RDFS file containing the application domain ontology (domain.rdfs) and a file containing the ontology domain instance (dataDomain.rdf).

The **Application Domain Ontology** is extracted from the following products of the Conceptual Design:

– **The Structural Diagram** allows us to express concepts and relationships that appear in the application domain by means of composite nodes which are connected to their simple or composite components by means of two abstraction mechanisms: *aggregation*, which refers to a set of nodes as a whole and *generalization*, which represents an inclusion relation involving inheritance mechanisms.

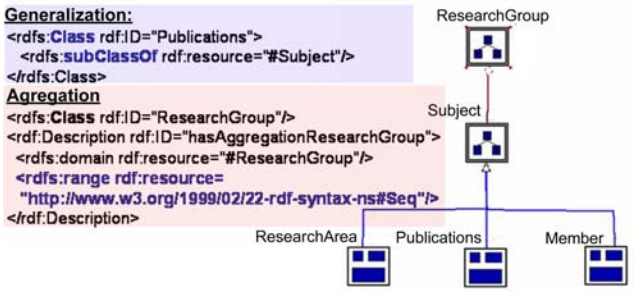


Figure 3. Structural Diagram from AriadneTool

On the proposed example we want to represent a research group composed of members, research areas and publications. To represent it on AriadneTool, the designer draws the general structure composed of a node Subject that is an aggregation of the node ResearchGroup and the nodes Member, Research and Publication that are a generalization of the node subject. This representation is shown on the right side of Figure 3. On the left side, Part of the Application Domain Ontology that is extracted automatically from the design is shown. On this ontology *generalization* is represented with the property: subClassOf and *aggregation* with a new property with a range which is a sequence.

– **The Navigation Diagram** specifies the navigation paths and tools that the website is going to offer to the users. Navigation paths are settled among nodes using tagged links which can be uni or bi-directional. In the *Member* node example we browse both the *Publications* and the *ResearchArea* node. Since *Subject* is a generalization composite, all its components (such as *Member*, *Publication*, and so on) will inherit the link information.

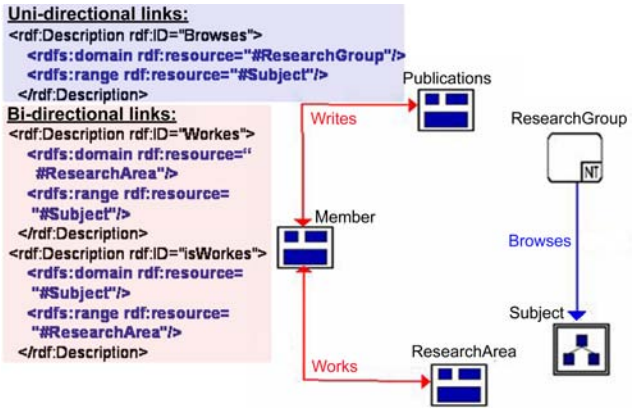


Figure 4. Navigation Diagram from AriadneTool

Right side of Figure 4 shows a screenshot of the drawing of the Navegation Diagram captured from AriadneTool. The left side contains part of the Application Domain Ontology that is extracted automatically from the design. On this ontology uni-directional links are represented with a property whose domain is the node source and the range is the target node. Bi-directional links are represented through two properties, alternating source and range.

– **The Internal Diagram** stores information about the spatial as well as temporal dimension of each of the information elements identified in the structural and navigation diagrams.

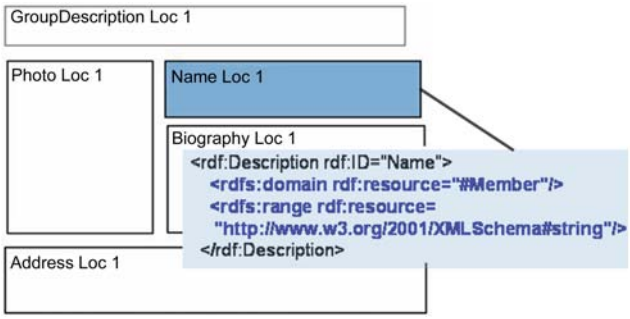


Figure 5. Spatial Diagram from AriadneTool

The left side of figure 5 shows the Member node visualization area with its contents represented with white boxed, located and aligned. From an ontological point of view, contents are like ontology properties, they are defined in an independent way and are then tied to different nodes, and thus, they can have different domains. For example, the properties of a group Member

include its photo, name biography and address. On the ontology fragment presented on the right side of Figure 5 each node content is presented as a property whose domain is the node where it is included and the range is a reference to the information that will be included in the node.

The ontology domain instance will be extracted in the next stage, the *Detailed Design* where the entities specified in the *Conceptual Design* are transformed into more concrete system elements. It is extracted from the following products:

```
<j.0:Member rdf:ID="id73324">
<j.1:NameType rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Tex
</j.1:NameType>
[...]
<j.1:NameX rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">237
</j.1:NameX>
</j.0:Member>
```

Figure 6. Domain ontology metadata

– **The Diagram of Nodes Instances** where the nodes defined in the structural diagram are created by means of a number of Node Instances. Thus, the Member node is replicated as many times as needed to represent all group members.

– **The Detailed Internal Diagrams** where all nodes and contents are fully specified and annotated with their values.

Figure 6 is a part of the RDF/XML encoding for the research group where the instances of Member node appear with its values attached to the ResearchGroup resource. From these metadata described in RDF and the structure of the application domain ontology described in RDFS and generated as explained on previous section, we can develop a semantic web application.

4.2 The presentation view

The application domain ontology and its instances enable us to define concepts, relations and system data information to be processed by software tools. However we need to store information on presentation in order to show the semantic content on a conventional browser. This information is extracted from two detail levels:

– **Hypermedia information relative to the model** annotate hypermedia content according to the Labyrinth model. On Figure 7(a) we can see a fragment from the Labyrinth model ontology and the data extracted from the research group example. This is a high level description extracted from the different products of the Conceptual Design such as the *User Diagram* and the *Access Table* (Figure 8) in which the designer can define presentation rules. Using this information we can decide which user accesses each node (content). Taking as example the knowledge presented in Figure 7(a) we can think about the following question: what user or users has priviledge to reach the photo content of the Member node? or what content belongs to each node. This information is extracted from the sematic repository containing ontologies and data, using RDQL as query language as mentioned on section 4.

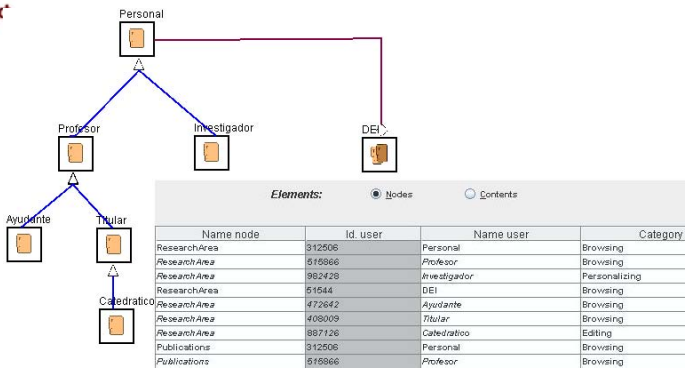


Figure 8. User Diagram and Access Table of AriadneTool

– **Presentation details** is a low level description of the layout, size and content type. First we generate an ontology about the presentation and then we extract the data from the Internal Diagram of the Conceptual Design (Figure 7(b)).

5. Related works

Currently, other model-driven approaches such SHDM [10], WSDM [14], OOWS [5] or UWE [9], follow the same strategy as ADM with some differences. As mentioned on previous sections, these approaches use Semantic Web-based languages (e.g. OWL, RDFS, RDF) to specify those relevant conceptual constructs that characterize the meaning of the corresponding Web Application. This specification makes possible the connection of the application to any external potential agent. This way, models can be represented by Semantic Web languages. ADM is different from other methods in the sense of implementation of security mechanisms defining users access policies. It also integrates information relative to the navigation into the presentation high level ontology. On this ontology, ADM stores the naviga-

Fragment of labyrinth.owl

```
<owl:Class rdf:ID="Simple">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Node"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#ContainerLaby"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Composite">
  <rdfs:subClassOf rdf:resource="#Node"/>
  <rdfs:subClassOf
    rdf:resource="#ContainerLaby"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="hasContent">
  <rdfs:range rdf:resource="#ContainerLaby"/>
  <rdfs:domain rdf:resource="#Location"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasAccess">
  <rdfs:range rdf:resource="#User"/>
  <rdfs:domain rdf:resource="#ContainerLaby"/>
</owl:ObjectProperty>
<owl:Class rdf:about="#User">
  <rdfs:subClassOf
    rdf:resource="#HMElement"/>
</owl:Class>
```

Fragment of dataLabyrinth.rdf

```
<j.0:Simple rdf:ID="Member"
    j.0:hasAccess="Personal"
    j.0:hasContent="Photo">
  <j.0:hasAccess>Profesor</j.0:hasAccess>
  <j.0:hasAccess>DEI</j.0:hasAccess>
  <j.0:hasAccess>Ayudante</j.0:hasAccess>
  <j.0:hasAccess>Titular</j.0:hasAccess>
  <j.0:hasContent>Name</j.0:hasContent>
  <j.0:hasContent>Biography</j.0:hasContent>
  <j.0:hasContent>Address</j.0:hasContent>
</j.0:Simple>
<j.0:Composite rdf:ID="Subject"
    j.0:hasAccess="Personal"
    j.0:hasNodeGeneralization="ResearchArea"
    j.0:hasContent="SubjectElement"
    j.0:hasLink="Browses">
  <j.0:hasAccess>Profesor</j.0:hasAccess>
  <j.0:hasAccess>DEI</j.0:hasAccess>
  <j.0:hasAccess>Ayudante</j.0:hasAccess>
  <j.0:hasAccess>Titular</j.0:hasAccess>
  <j.0:hasNodeGeneralization>Publications
  </j.0:hasNodeGeneralization>
  <j.0:hasNodeGeneralization>Member
  </j.0:hasNodeGeneralization>
</j.0:Composite>
```

(a) The Labyrinth model ontology and data

Fragment of presentation.rdfs

```
<rdf:Description rdf:ID="NameType">
  <rdfs:domain
rdf:resource="dominio.rdfs#Member"/>
  <rdfs:range rdf:resource=
"http://www.w3.org/2001/XMLSchema#string"/>
</rdf:Description>
[...]
<rdf:Description rdf:ID="NameX">
  <rdfs:domain rdf:resource=
    "dominio.rdfs#Member"/>
  <rdfs:range rdf:resource=
    "http://www.w3.org/2001/XMLSchema#int"/>
</rdf:Description>
```

Fragment of dataPresentation.rdf

```
<j.0:Member rdf:ID="id73324">
<j.1:NameType rdf:datatype=
"http://www.w3.org/2001/XMLSchema#string">
  Text
</j.1:NameType>
[...]
<j.1:NameX rdf:datatype=
"http://www.w3.org/2001/XMLSchema#integer">
  237
</j.1:NameX>
</j.0:Member>
```

(b) Presentation ontology and data

Figure 7.

tional features of web applications and provides information about how data can be accessed by others.

6. Conclusions

This paper has argued how hypermedia design methods can provide semantic contents as well as contextual information about the application domain which are modeling in order to face up the Semantic Web. Mapping models to an ontology language provides us some benefits such as the decrease of the cost and risk of the application design and information sharing between applications. We incorporate semantic content generation using the Ariadne Method. We apply this approach on AriadneTool.

Web designers will now provide the annotation during the Conceptual Design. Compared to currently existing annotation methods, this approach extracts the semantic content implicitly so the designer does not realize the process; no additional expert knowledge is necessary for the data annotation and the ontology and data generated. Finally, this process enables us to improve the consistency during the application design process and to speed it up by making use of the metadata already provided. Also we can browse semantic content on a conventional browser using the visu-

alization module that complements AriadneTool. Finally this approach establishes a technological framework where the application data and functionality can be presented and shared between different web applications.

To conclude, we are extending the architecture presented here for sharing and reusing semantic content generated in the design process of other applications. In addition we are extending the functionality of AriadneTool for importing ontologies already defined as does HERA [16] and OntoWebber [8]. This will allow to validate designs or begin the design from domains previously defined adopting those concepts that are of utility.

7. Acknowledgments

This work is part of the ARCE++ project (TSI2004-03394) funded by the Spanish Ministry of Science and Education (MEC) and a cooperation agreement between "Universidad Carlos III de Madrid" and "Dirección General de Protección Civil y Emergencias" (Ministry of the Interior).

References

[1] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, Mayo 2001.

[2] S. Ceri, P. Fraternali, and A. Bongio. Web modeling language (WebML): a modeling language for designing web sites. *WWW9 / Computer Networks*, 33(1-6):137–157, 2000.

[3] P. Díaz, I. Aedo, and S. Montero. Ariadne, a development method for hypermedia. In *proceedings of Dexa 2001*, volume 2113 of *LNCS*, pages 764–774, 2001.

[4] P. Díaz, I. Aedo, and F. Panetsos. Labyrinth, an abstract model for hypermedia applications. description of its static components. *Information Systems*, 22(8):447–464, 1997.

[5] J. Fons, V. Pelechado, M. Albert, and O. Pastor. Development of web applications from web enhanced conceptual schemas. In *Proc. Of the International Conference on Conceptual Modelling, 22nd Edition, ER'03*, 2003.

[6] G. Guizzardi, H. Herre, and G. Wagner. On the general ontological foundations of conceptual modeling. In *21st International Conference on Conceptual Modeling (ER2002)*, volume 2503 of *LNCS*, pages 65–78, 2002.

[7] T. Isakowitz, E. A. Stohr, and P. Balasubramanian. RMM: a methodology for structured hypermedia design. *Comm. of the ACM*, 38(8):34–44, 1995.

[8] Y. Jin, S. Decker, and G. Wiederhold. Ontowebber: Model-driven ontology-based web site management. In *The 1st International Semantic Web Working Symposium SWWS'01*, 2001.

[9] N. Koch, A. Kraus, and R. Hennicker. The authoring process of the uml-based web engineering approach. *In First International Workshop on Web-Oriented Software Technology*, 2001.

[10] F. Lima and D. Schwabe. Application modeling for the Semantic Web. In *Proceedings of the First Latin American Web Congress (LA-WEB'03)*, pages 93–102. PUC-RIO, IEEE Computer Society, 2003.

[11] D. Mcguinness, R. Fikes, J. Hendler, and L. Stein. DAML+OIL: an ontology language for the Semantic Web . *IEEE Intelligent System*, 5(17):72–80, 2002.

[12] S. Montero, P. Díaz, and I. Aedo. Toward hypermedia design methods for the semantic web. In *14th International Workshop on Database and Expert Systems Applications (DEXA'03)*, pages 762–767. IEEE Computer Society, 2003.

[13] S. Montero, P. Díaz, and I. Aedo. Ariadnetool: A design toolkit for hypermedia applications. In *Journal of Digital Information*, 2004.

[14] P. Plessers and O. D. Troyer. Annotation for the semantic web during website development. In *Web Engineering - 4th International Conference, ICWE 2004, Munich, Germany, July 26-30, 2004, Proceedings*, pages 349–353, 2004.

[15] V. Torres, J. Fons, V. Pelechano, and O. Pastor. Navigational modeling and the semantic web. an ontology based approach. In *WebMedia & LA-Web 2004 Joint Conference 10th Brazilian Symposium on Multimedia and the Web 2nd Latin American Web Congress*, pages 94–96. IEEE Computer Society, 2004.

[16] R. Vdovjak and G.-J. Houben. Rdf-based architecture for semantic integration of heterogeneous information sources. In *Workshop on Information Integration on the Web*, 2001.

Reusable Navigation Templates to Support Navigation Design in Hera

Peter Barna, Geert-Jan Houben, Ad Aerts, Flavius Frasincar, and Philippe Thiran
Technische Universiteit Eindhoven
PO Box 513, NL-5600 MB Eindhoven, The Netherlands
{p.barna, g.j.houben, a.t.m.aerts, f.frasincar, ph.thiran}@tue.nl

Abstract

Reuse is a fundamental concept in software design. It has many aspects and can be applied at various levels of abstraction. In this paper we focus on the reuse of high-level (design model) specifications of software components in the design of web applications. Concretely, we discuss the reuse of navigation templates to specify (parts of) navigation models in different application domains based on different data sources. While supporting this diversity of applications, at the same time navigation templates should allow easy deployment. In this paper we propose a solution to this apparent contradiction using a component-specific conceptual model. By applying a mapping from this model to a concrete domain model, an automatic deployment of the navigation templates can be performed. The process of navigation template design and deployment (including the process of defining the mapping) is explained and demonstrated on two examples using the Hera framework and its HPG software.

1 Introduction and Related Work

One of the major concepts in software design is the reuse of software artifacts applied at different levels of abstraction - from reuse of system requirements to reuse of software code, and at different levels of granularity - from reuse of software packages or whole applications through use of design patterns [4] to reuse of classes (concepts) organized in hierarchies.

The benefits of reuse are obvious, and include saving software development effort (avoiding redundant design), facilitating the maintenance of software systems, and making the design traceable and transparent.

Due to the specific nature of the Web, extensions of traditional software design methods have been proposed for the development of web applications. The support of navigation and the necessity of navigation modelling is what distinguishes web systems and web design methods from traditional systems and design methods. In the navigation modelling, reuse of navigation structure specification (often referred to as "Web Patterns" or "Web Design Patterns") is a very useful technique, but its full exploitation is still an open problem. A good overview of common web patterns is presented in [11]. The descriptions of patterns presented there serve as a set of handy guidelines for web designers. Existing software libraries offer a wide variety of useful generic primitives that can be (re)used during the construction of web applications, but they rarely contain larger navigation patterns. Since navigation models are usually tightly coupled with concrete domains, specified by Conceptual Models (CM), the achievement of the domain portability is not a trivial task.

Current methods for web design already benefit from the reuse concept in various ways. WebML [3] specifies the navigation structure by means of different (predefined) types of units. The method allows easy and convincing composition of different units. Most of the units however are associated with concrete data (specified in a data model), so using such composed patterns for different domains is not trivial. Object-oriented approaches like OOH [5], UWE [7], or OOWS [10] show a solid approach supporting object-oriented reuse techniques like class abstraction. The problem of domain portability of navigation models in object-oriented environments using the OOHDM method is discussed in [13]. The web design framework introduced there represents abstract navigation models that are isolated from concrete domains and can be instantiated to a concrete domain. The deployment process consists of the derivation of a concrete OOHDM model from an OOHDM-frame (a generic conceptual model). It is possible to build common navigation patterns, however the navigation classes are derived from conceptual classes describing a concrete domain.

In this paper we propose a practical approach for the design and deployment of domain portable reusable navigation patterns called Navigation Templates (NT). We focus on the explanation of the mapping from a Template Conceptual Model (TCM) describing the structure of data used within an NT, to a concrete domain conceptual model (CM).

For an NT, such mapping is defined for every concrete domain, and it is a kind of NT parametrization. It allows not only a relatively easy deployment of an NT to a concrete domain, but it also facilitates the specification of possible data manipulations in an NT. A mapping is similar to a data (schema) integration model, and we can benefit from existing knowledge in this field of research. The process of deployment of such an NT to a concrete domain can be automated by using an NT specification and an appropriate mapping to a concrete domain. This deployment process is demonstrated on two examples using the Hera framework [6, 16]. Despite the use of a concrete method for reasons of illustration the proposed approach of mapping NTs to concrete domains can be used for other methods.

Section 2 explains the requirements for NTs and the context of their usage. The core of the paper is Section 3 that explains the approach in detail using two examples in Hera and HPG (Hera Presentation Generator web server software). Potential mapping (data integration) problems and solutions are also discussed here. The current work on the creation of software tools supporting the design and deployment of NTs for Hera is briefly explained in Section 4 and the text is concluded by Section 5.

2 Navigation Templates Overview

A Navigation Template (NT) is a parameterized conceptual specification of a navigation structure. This specification has well-defined interfaces in terms of links and types of information they can carry. The NT parametrization allows its deployment within existing navigation models of similar applications based on different data domains. An example of a primitive NT is a user-selection device (a virtual shopping basket) that is used in a web application for an online sport equipment shop as a classical shopping basket, and in a university online library it can be used for instance as a tool facilitating the searching of publications by choosing topics of interest.

Besides the navigation structure, NTs define also some basic application logic (functionality), in most cases related to dynamic updates of the navigation structure and underlying data, both based on the user interaction. Although NTs represent conceptual reusable units, we demonstrate how they can be converted to a specification that is directly used by a web server software to provide desired functionality on the Web.

For the generation of a deployed NT we need two kinds of specification:

- An **NT specification** that contains two sub-models:
  - The **Template Conceptual Model** (TCM) describes the structure of data concepts (and their concept relationships) that are used in the description of the NT's navigation structure (TAM). Note that the content domain described by a TCM is not necessarily materialized, but the TCM is used in the parametrization when the NT is deployed.
  - The **Template Application Model** (TAM) describes the navigation structure of the NT and its application logic. It is based on the data defined in a TCM.
- An **NT parametrization** that defines the mapping from the TCM to a concrete domain. This mapping allows a (semi-)automatic translation of the NT into (parts of) a concrete navigation model.

With a certain level of abstraction we can see an analogy between NTs and MDA [9] models. A main concern of MDA is the support for the design and integration of systems based on different platforms. In the same spirit, NTs represent domain independent models, so if we replace the notion of platform with the notion of domain, a PIM (Platform Independent Model) of MDA can be compared to a domain independent NT specification, and PSM (Platform Specific Models) to concrete deployed NTs. Analogically, domain independent NTs can be (semi)automatically deployed for different domains and frameworks using different transformation (deployment) tools.

2.1 Benefits of NT Reuse and Methodological Issues

The main benefit of building NTs and their later deployment lies in saving development effort for parts of web applications that can be used again and again for different domains. An interesting application of NTs is the composition of new web applications from an available NT on already existing domains (for instance on legacy databases). Figure 1 sketches how NTs can be deployed within a navigation model. The thick arrows represent hyperlinks (possibly) carrying parameters (the depicted internal structure of the NT does not reflect any real structure and is sketched only for illustration purposes). The thin arrows show the deployment process with the transformation of an NT specification to a concrete (part of) navigation model based on a CM. This transformation is automatic, but uses a manually built TCM-to-CM mapping. The situation in Figure 1 requires two mappings, since the same shopping basket NT is used for different data concepts (though within the same (larger) domain). The real benefit of NT as a conceptual unit of reuse depends on several aspects including:

- generality of NT design, in the sense of how easily they can be used for different applications. This is in-

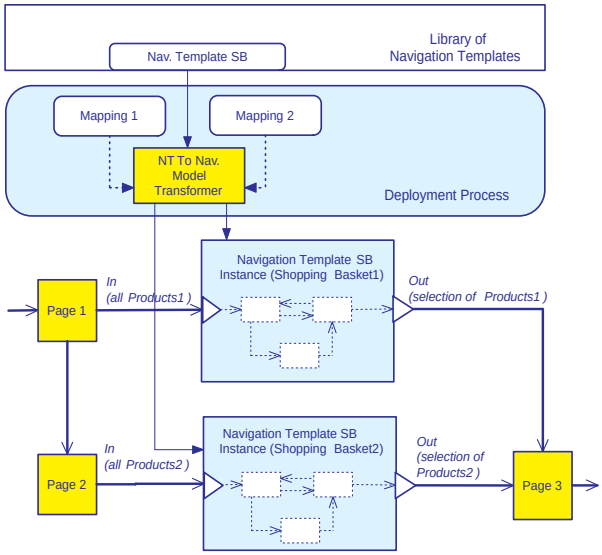


Figure 1. NT deployment example

fluenced by a good selection of ”typical” web application patterns NTs represent, and also by minimizing the structure of NTs (smaller and simpler units are easier to deploy and specialize).

- complexity of NT deployment, including:
  - complexity of its parametrization. That can be influenced by proposing simple TCMs with a minimal number of mappings to a concrete domain. The mapping specification can be facilitated by design tools.
  - automation of the transformation of the NT and appropriate mapping to a concrete models and/or executable specification eventually. This is given by availability of appropriate software translators.

In the context of the NT design process, we consider two possible types of the design cycle:

- **Data-driven design**, where first the TCM (or a set of TCMs) is defined and then the TAM is built on top of it (in other words, first the data, then the navigation). This approach can be used when a simple and straightforward NT parametrization (mapping to a concrete domain) is vital. For instance, when we want to use concrete, complex legacy databases, TCMs are better starting points. A concrete web application based on such a legacy database then can be easily composed from already built NTs, because the structures of the concrete TCM are in accordance with the CM at hand.
- **Process-driven design**, where on the basis of user requirements, a process model is defined for an NT and then subsequently enriched with appropriate data model and data manipulation specifications (in other words, first the process, then the navigation). From the data model an NT specification can be derived. This interesting, process-driven approach is part of ongoing work and will be described in a separate paper.

The details of the NT specification and deployment techniques may depend on the concretely used approach. In the following text we explain the principles of NT specification and deployment using two examples. The first example demonstrates a multiple use of a simple NT (in this case a guided tour) in a single application, and the second highlights potential problems associated with mapping a TCM to a concrete domain and their resolution.

3 Navigation Templates in Hera

For a better explanation of the NT concept, we demonstrate its basic principles using the Hera framework and two examples. In the Hera design cycle, an NT can be generated from a more abstract process model, or it can be designed manually. The role of NTs in the method and its models is depicted in Figure 2. It has already been mentioned, that an NT contains a TCM describing the structure of the information that will be presented and processed, and it contains an appropriate TAM specifying the navigation view on the TCM. The *Articulations* (see [16]) represent the NT parametrization - the mapping from the TCM to a concrete CM, i.e. the ”binding” of the TCM to the concrete domain. The NT specification together with the *Articulations* are used by the *NT2AM Transformer* to generate a concrete (part of an) AM describing the navigation structure and functionality of a concrete web application. The AM can then directly be used by a Hera engine such as HPG-Java for the online generation of pages in the web application.

3.1 Brief Overview of Hera

Within the Hera project we investigate methods for the specification of (dynamic) hypermedia presentations and we build and maintain appropriate server software and design tools. The methodology determines a number of design steps resulting in a set of models (that specify how the hypermedia presentations get generated). The conceptual design phase results in a Conceptual Model (CM) defining the structure of source data used in presentations. The application design phase results in an Application Model (AM) defining a navigation structure over the CM, possibly with data manipulation associated with user actions. The presentation design phase produces a Presentation Model (PM) specifying the layout of presentations. All Hera models are expressed in RDFS [2].

These models are used by a Hera engine, in this case HPG-Java (a software module running as a servlet hosted by a web server), first performing data retrieval, and then performing data transformations resulting in presentation pages, possibly for different platforms and different formats (HPG supports HTML, WML, and SMIL for presentations without data manipulation, and HTML for presentations allowing forms and data manipulation). The bottom part of Figure 2 shows the transformations of the Hera/HPG pipeline, where retrieved data is transformed consecutively to a CM instance (CMI), an AM instance (AMI), and a presentation in a concrete format (e.g. HTML). All intermediate data chunks are internally represented in RDF [8].

The application modelling method is capable of expressing more advanced functionality than only a navigation view over static data content. It supports modelling of user inputs by means of forms allowing users to enter arbitrary information possibly exploited by data manipulation queries. All queries in Hera AM are expressed in the SeRQL [1] RDFS query language with slight modifications (queries are pre-processed by the Hera engine) including the management of session parameters (variables).

An AM contains basic building blocks called slices that describe the structure of navigation pages (or their parts since they can be nested), and their linking (see Figure 3). Slices can have root concepts (from the CM) drawn as large ovals in the slice upper part. If a slice does not have the root concept, it is a constant slice and it can have arbitrary content. If the target of a link is a non-constant slice, the link carries parametrization that determines the instantiation of the target slices (the anchor determines what instance of the target slice root concept is used for the target slice instantiation). The instantiation of slices can be determined also by queries associated with slices. A slice can contain attributes (literal properties in the CM) from a root concept and attributes from concepts related to the root concept (the bottom part of the slice shape) connected with the root concept by CM properties. User interaction is facilitated by the use of forms that carry a number of input fields users can fill in. Forms have associated processing queries that can retrieve data, change the data content, or update values of session variables.

3.2 Guided Tour Example

In the first example we demonstrate the multiple use of a very simple NT. The NT represents a guided tour - a step-by-step (one per web page) presentation of multiple concept instances. This concrete NT we deploy twice in a simple museum application. Once for the presentation of painters, and once for the presentation of paintings. For every concrete deployment we will show a set of articulations (mapping the TCM to a CM).

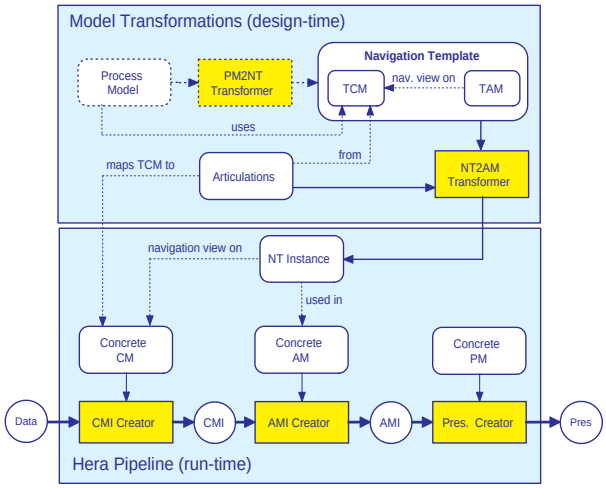


Figure 2. Role of NTs in Hera architecture

Figure 3 presents the TCM and TAM of the guided tour NT. It uses a special predefined sub-class of general slice (iterator) that allows easy implementation of a guided tour. A slice of type *Iterator* comes with a default *IteratorForm* providing a navigation facility through a collection of the *Item* instances and allowing to exit the iteration using the *Out* button. In the case of exit the instance of the last viewed *Item* is provided as the output parameter. The concrete data source containing information about painters and paintings is specified by its CM shown in Figure 4.

Figure 5 shows an AM of a simple museum application using two instances of the Guided Tour NT. They contain attributes based on a mapping from the TCM to the CM and also attributes not appearing in the original NT, but added by the designer. The application presents a set of painting techniques (the *TechniquesList* slice), for every technique a list of paintings exemplifying the technique (the *Technique.PaintingList* slice), and the two guided tours for browsing through the painters and paintings (*GD-Painters* and *GD-Paintings*).

3.3 Publications Example

In the second example we demonstrate possible problems with mapping a TCM to a concrete CM. To a large extent, this is a problem of data integration. We show some typical cases appearing in RDFS schema integration and present some practical solutions covering the needs of our method (including the rewriting of selection and data manipulation queries). Most of the problems have been recognized and studied in [12, 14]. The example application is a publication database that stores publications, authors (researchers), and research groups. The Publications NT is specified in Hera and later deployed to an existing data

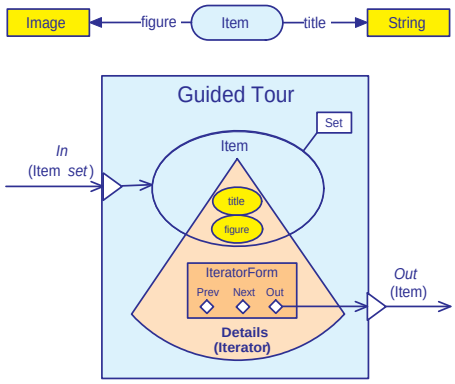


Figure 3. Guided Tour NT, the TCM on top, the TAM below

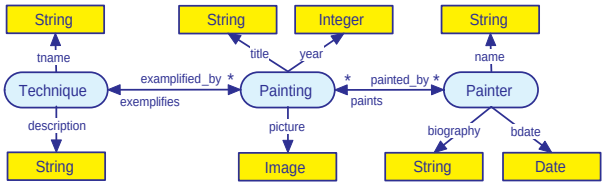


Figure 4. CM of a museum application

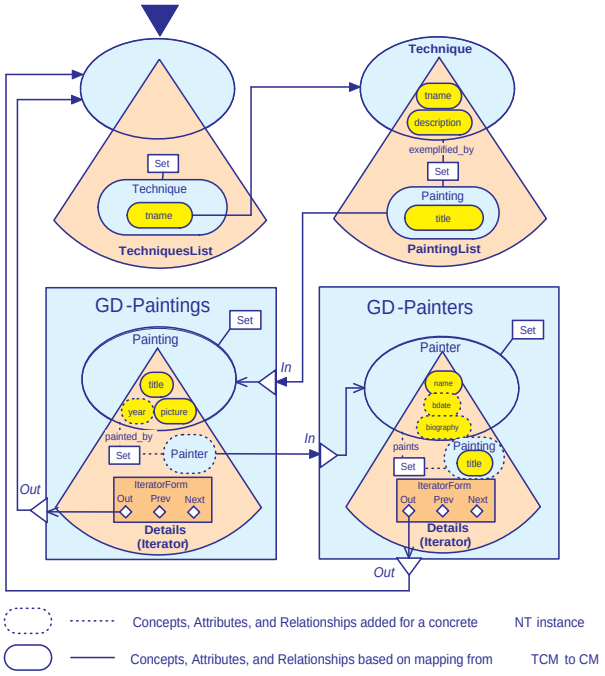


Figure 5. AM of a museum application deploying instances of the Guided Tour NT

source with a different data structure. A set of articulations is defined.

3.3.1 Template Conceptual Model

The TCM contains only those concepts, concept properties, and literal properties that are necessary for describing the core navigation structure and functionality (we show only the addition of a publication) associated with a concrete NT. The TCM of the Publications NT is presented in Figure 6.

3.3.2 Template Application Model

Unlike the first example, here the TAM also contains the specification of input forms and their processing. The Publications TAM shown in Figure 7 consists of three slices presenting research groups (the *Group* slice), listing researchers in groups (the *Group.Researcher* slice), and listing publications of a researcher (the *Researcher.Publications* slice). The *AddPaper* slice allows adding a publication created by a concrete researcher (only one in our example). For the sake of simplicity the application has not been detailed completely. We omit here more comprehensive data manipulations, like removing publications, adding a new researcher, and managing groups. The *AddPaper* query is activated when the *AddPaper* form is submitted:

```
CONSTRUCT DISTINCT
   {P}rdf:type{tcm:Paper};
      tcm:ptitle(Title);
      tcm:url{URL};
      tcm:published_at{Published};
      tcm:author{Author};
      tcm:year{Year}
FROM
  {form:AddPaper}<form:Iptitle>{Title},
  {form:AddPaper}<form:Iurl>{URL},
  {form:AddPaper}<form:Ipub>{Published},
  {session:session}<session:resID>{Author},
  {form:AddPaper}<form:Iyear>{Year}
```

In this query a new instance of a paper is created where its properties are taken from the *AddPaper* form inputs. The value assigned to the *Author* variable represent the ID of the last presented researcher (refreshed by the *SetResearcher* query during the *AddPaper* slice instantiation and temporarily stored in the *session:resID* session parameter). The *SetResearcher* query is very simple:

```
SELECT
  R
FROM
  {session:session}<session:sliceid>{R}
```

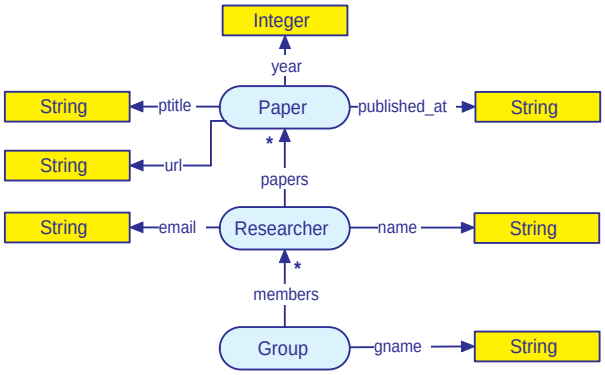


Figure 6. Publications TCM

The *session:sliceid* is a default session variable containing the URI of the root concept instance of the last completely instantiated slice (that is why it is attached to the *AddPaper* slice and not to *Researcher.Publications* slice, although it contains the URI of the current *Researcher*). The value of the *R* variable is in the RDFS TAM specification assigned to *session:resID*. The complete query and session parameter specification does not appear in the TAM diagram, but it is in the TAM RDFS file:

```
<rdfs:Class rdf:ID="SetResearcher"
            slice:execute="Once">
   <rdfs:subClassOf rdf:resource=
      "http://wwwis.win.tue.nl/
      ~hera/ns/slice#Query"/>
   <slice:queryString>
      SELECT R
      FROM {session:session}
              <session:sliceid>{R}
   </slice:queryString>
</rdfs:Class>

<rdfs:Class rdf:ID="QueryResult_ID101"
            slice:resultName="resID"
            slice:useAsSessionVar="Yes">
   <rdfs:subClassOf rdf:resource=
      "http://wwwis.win.tue.nl/
      ~hera/ns/slice#QueryResult"/>
</rdfs:Class>
```

3.3.3 Mapping NTs to Concrete Domains

A necessary condition for the automated transformation of an NT to an AM for a concrete CM is the existence of a mapping from the abstract TCM to a concrete domain model. We demonstrate the specification of such a mapping using the Publications example and show possible situations. Figure 8 presents a concrete CM describing a do-

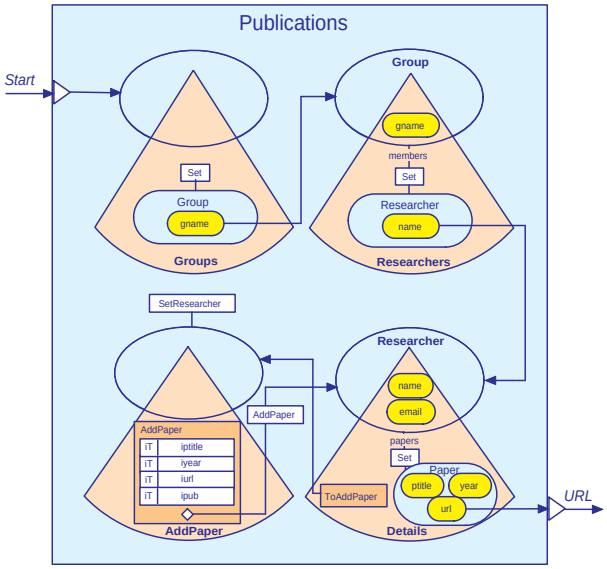


Figure 7. Publications TAM

main of publications. There are two categories of mapping. The first one is concept-to-concept, which facilitates the determination of root concepts and data manipulation queries during transformation of the TAM slices into concrete AM slices. The second one is attribute-to-attribute, which allows the transformation of slice attributes and is used in query transformations as well. We now define mappings of type concept-to-concept and attribute-to-attribute. We use path expressions specifying concept-property chains in the form $\{Concept1\}property1\{Concept2\}$.... Inverse properties are denoted as $\{Concept\}property^{-1}$. If the value of a TCM property is constructed from the values of several properties in the CM (concatenation), we write it as $\{Concept1\}property1 \odot \{Concept2\}property2$. The fact that the value of a TCM property is retrieved from several CM properties is captured as $\{Concept1\}property1 \cup \{Concept2\}property2$. In this case the mapping is a union of values of the given path expressions. The mapping of concepts relies on the uniqueness of the concept names (in other case we would need to use path expressions as well), in our example the mappings for the concepts are:

- for *tcm:Group* no mapping is defined
- *tcm:Researcher* is mapped to *cm:Person*
- *tcm:Paper* is mapped to *cm:Paper*

For the mapping of attributes we define articulations containing pairs of path expressions for the TCM and the CM. The mapping is described in Table 1. Further details of the mappings are explained in Section 3.4.

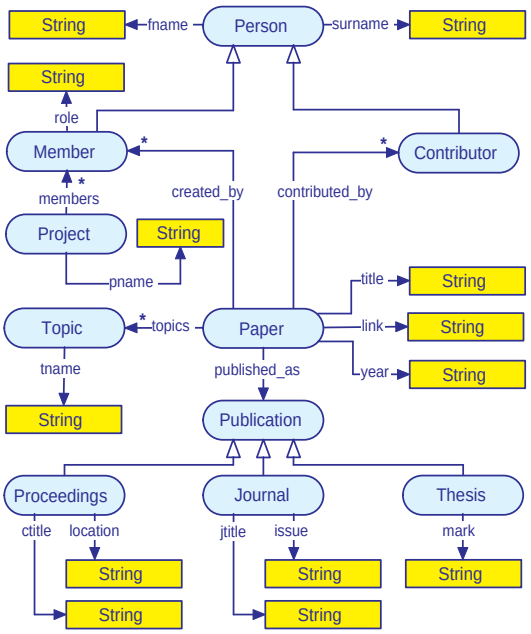


Figure 8. Concrete Publications domain CM

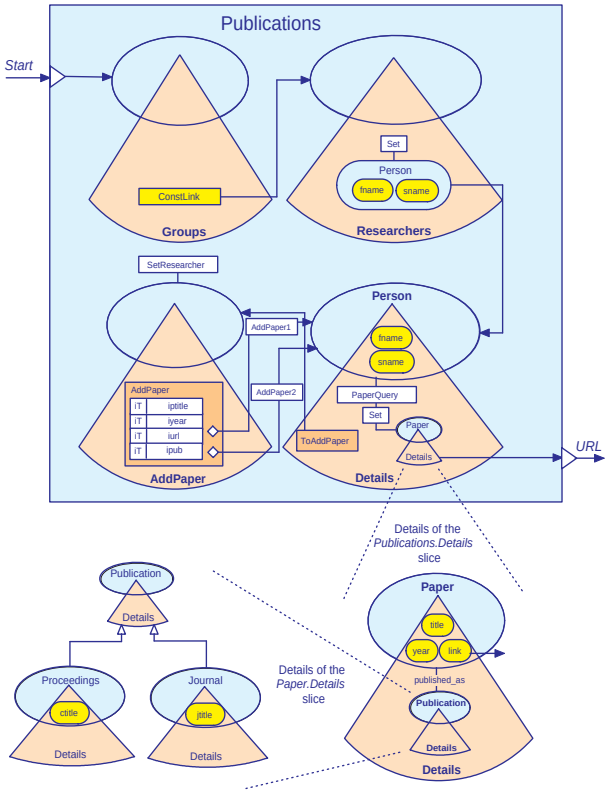


Figure 9. Deployed Publications NT

Attribute in TCM	Mapping to CM
$\{Group\}gname$	no appropriate range concept in CM, and thus no mapping. In CM it will be represented by a constant string (a name of a working group)
$\{Researcher\}name$	$\{Person\}fname \odot \{Person\}surname$
$\{Researcher\}email$	no appropriate attribute in CM
$\{Researcher\}papers$	$\{Member\}created\_by^{-1} \cup \{Contributor\}contributed\_by^{-1}$
$\{Paper\}ptitle$	$\{Paper\}title$
$\{Paper\}published\_at$	$\{Paper\}published\_as \{Proceedings\}ctitle \cup \{Paper\}published\_as \{Journal\}jtitle$
$\{Paper\}year$	$\{Paper\}year$
$\{Paper\}url$	$\{Paper\}link$

Table 1. TCM to CM attribute mapping

3.3.4 Deployed Navigation Template

When the *Articulations* are specified, an appropriate deployed NT can be generated. A deployed NT is an (part of) AM. In this process slice relationships based on the TCM are replaced by those based on the CM at hand. Due to possibly different schema structures of a TCM and a concrete CM, in some cases simple slice aggregations based on a single CM relationship must be replaced by more complex queries. For instance, this is the case when for a path expression in the TCM with the length (number of properties in the path expression) one, there exists a corresponding path expression in the CM with length more than one (the result would be a join query). During the deployment process the *papers* slice aggregation in the original TAM *Researcher.Details* slice is automatically transformed to a query (we name it here *PaperQuery*) that is a union of the two queries (for the *Person* subclasses *Member* and *Contributor*):

```
SELECT X
FROM {P}contributed_by{X}
```

and

```
SELECT X
FROM {P}created_by{X}
```

where *P* is an instance of the *Person* concept given by the *Person.Details* slice instance.

The *AddPaper* query appearing in the original TAM is during the deployment process automatically transformed into two different queries, one for the *Proceedings* subclass of *Publication*:

```
CONSTRUCT DISTINCT
   {P}rdf:type{cm:Paper};
      cm:created_by{M},
   {Proc}rdf:type{cm:Proceedings};
      cm:ctitle{Title};
      cm:link{URL};
      cm:year{Year};
      cm:ptitle{Published},
   {P}cm:published_at{Proc}
FROM
   {session:session}<session:resID>{M},
   {form:AddPaper}<form:Iptitle>{Title},
   {form:AddPaper}<form:Iurl>{URL},
   {form:AddPaper}<form:Ipub>{Published},
   {form:AddPaper}<form:Iyear>{Year}
```

and one for the *Journal* subclass of *Publication*:

```
CONSTRUCT DISTINCT
   {P}rdf:type{cm:Paper};
      cm:created_by{M},
   {Proc}rdf:type{cm:Journal};
      cm:title{Title};
      cm:link{URL};
      cm:year{Year};
      cm:jtitle{Published},
   {P}cm:published_at{Proc}
FROM
   {session:session}<session:resID>{M},
   {form:AddPaper}<form:Iptitle>{Title},
   {form:AddPaper}<form:Iurl>{URL},
   {form:AddPaper}<form:Ipub>{Published},
   {form:AddPaper}<form:Iyear>{Year}
```

The queries create different *Publication* types (subclasses) with different attributes. The user of the application decides what kind of *Publication* he wants to add. This is facilitated by two buttons (one for each subclass and executing the first or the second query) that are automatically generated (during the NT deployment process) and placed to the *AddPaper* form (see Figure 9). Note the simplification we made here due to the lack of space. Both queries are applicable for the *Member* type of *Researcher* (can be determined by the *cm:created_by* property appearing in the CONSTRUCT clause). In a real example, every form button would execute two optional queries depending on the type of last visited *Researcher* (*Contributor* or *Creator*).

3.4 Major Problems in TCM to CM Mapping

We can highlight a few typical situations, where the mapping from the TCM to a CM is not as straightforward as for instance the naming conflicts naturally solved by paired path expressions explained in Section 3.3.3. In this section we mention the conflicts and their possible solutions. This descriptions are used as guidelines for developing the NT deployment software (*NT2AM Transformer* in Figure 2). Most of the possible situations have been discussed and classified in [14]. Concretely we name:

- Data representation conflict: corresponding literal properties in the TCM and a concrete CM have different data types. An example is the $\{Paper\}year$ property (*String* and *Integer* types).
- Missing literal property conflict: a TCM concept attribute does not have its counterpart in the CM. An example would be the $\{tcm:Researcher\}tcm:email$ attribute.
- Concept-property and property-concept conflicts can appear when a concept in the TCM is modeled as a (literal) property in the CM and vice versa.
- A few cases of schema isomorphism conflicts:
  - A TCM concept does not have its counterpart in the CM. An example would be the *tcm:Group* concept.
  - A TCM concept literal property has only a reversed counterpart in the CM. An example is $\{tcm:Researcher\}tcm:papers$ that can be mapped to $\{cm:Member\}cm:created\_by^{-1}$ (or to $\{cm:Contributor\}cm:contributed\_by^{-1}$).
  - A TCM literal property is mapped to (composed of) multiple attributes in the CM. An example is $\{tcm:Researcher\}tcm:name$ that is mapped into a concatenation of $\{cm:Person\}cm:fname$ and $\{cm:Person\}cm:surname$.
- Generalization conflicts where a TCM concept is mapped into a CM concept with more specializations. An example of this would be the $\{tcm:Paper\}tcm:published\_at$ literal property that can be mapped to $\{cm:Paper\}cm:published\_as\{cm:Proceedings\}cm:ctitle$ and to $\{cm:Paper\}cm:published\_as\{cm:Journal\}cm:jtitle$

We do not mention other possible conflicts such as integrity constraint conflicts that can arise when more sophisticated constraints are imposed on the concepts and their properties.

3.4.1 Data Representation Conflicts

In this case the data types of the corresponding literal properties are not compatible. A simple type conversion is made: concretely, the type of a conflicting TCM attribute is transformed (possibly during the model transformation

since we can transform schemas) to the data type of the corresponding CM attribute. Applied to our example, the type *Integer* of attribute $\{tcm : Paper\}year\_at$ in the TAM is changed to *String* in the resulting AM.

3.4.2 Missing Literal Property Conflict

In the case of such a conflict such a literal property (attribute) is omitted in the resulting concrete AM. An example is the *email* attribute of the *Researcher.Details* in TAM that does not appear in the resulting AM (see Figures 7 and 9).

3.4.3 Concept-property and property-concept conflicts

This conflict appears if a concept in the TCM is modeled as a literal property in the CM or vice versa. An example of the property-concept conflict is $\{tcm : Paper\}tcm : published\_at$ that is mapped to $\{cm : Paper\}cm : published\_as$. This conflict is discussed in Section 3.4.5.

3.4.4 A TCM Concept Does not Have a Counterpart in the CM

In this case the *NT2AM Transformer* must replace the missing concept with a single (virtual) constant concept, so all its attributes are constants. For instance, the example publication CM is intended for a single research group, so the group name will be replaced with a constant string. The replacement by a constant is needed due to the fact that some top-level slices can be based on non-existing concepts. During the transformation process these slices are replaced by constant slices.

3.4.5 Generalization Conflict

This problem typically occurs when the TCM concept has specializations with a different property structure. An example is the mapping of the $\{tcm : Paper\}tcm : published\_at$ attribute that can be mapped into $\{cm : Paper\}cm : published\_as\{cm : Proceedings\}cm : ctitle$, but also into $\{cm : Paper\}cm : published\_as\{cm : Journal\}cm : jtitle$ depending on the type of the publication (*Proceedings* or *Journal*).

The solution to this problem needs to cover the following two situations (as well as some other problems, but they appear to be simpler):

- Transformation of slices for presentation purposes (i.e. transformation of SELECT queries). In this case the result should be the union of two queries containing both path expressions.
- Transformation of data manipulation queries. For the data consistency reasons the type of the manipulated concept should be determined (especially when new instances are created), despite the fact that there is no notion of these specializations in the TCM. One of the possible solutions is the automatic generation of a selection input field allowing the user to choose the type of concept to be created (according to existing concept subclasses). In the example it would be a selection between the *Proceedings* and *Journal* concepts when adding a new publication.

3.4.6 A TCM Concept Property Has Only a Reversed CM Counterpart

This situation occurs when a TCM property does not have a directly matching counterpart in the CM, but there is a CM property with inverse semantics. There is no direct illustration of this in the example, but $\{tcm : Researcher\}tcm : papers$ can be mapped to the union of the inversions of $\{cm : Paper\}cm : created\_by$ and $\{cm : Paper\}cm : contributed\_by$.

3.4.7 A TCM Literal Property is Mapped to a Concatenation of Multiple CM Literal Properties

This is a case when an attribute is mapped to a concatenation of multiple literal properties. An example is the concatenation of $\{cm : Person\}cm : fname$ and $\{cm : Person\}cm : surname$ for the $\{tcm : Researcher\}tcm : name$. The solution is to replace one TAM slice attribute in the TAM by several attributes in the resulting AM.

4 Implementation

The usefulness of the approach described in this text relies to a large extent on the availability of tools supporting the design and automated deployment of NTs. The most essential tool is the *NT2AM Transformer* (Figure 2) that transforms an NT specification to a concrete Hera AM or a part of it using the mapping to a concrete domain CM. This tool is a single Java application that reuses some classes from the Hera Mediator [16] for the processing of articulations.

A design support tool for the graphical specification of mappings (articulations) is currently under development, and uses part of the functionality of the EROS RDFS Explorer [15] that provides an interface for building SeRQL queries, and thus also supports the building of path expressions, which are the essential part of articulations. It will allow rapid and easy specification of needed articulations. The main window of the tool is shown in Figure 10. The NT graphical design tools are based on existing Hera CM and AM Builders (for the construction of the TCM and TAM) that are also used for the regular (graphical) design of Hera applications (i.e. without using an NT). These tools are being updated for specification of the NT interfaces.

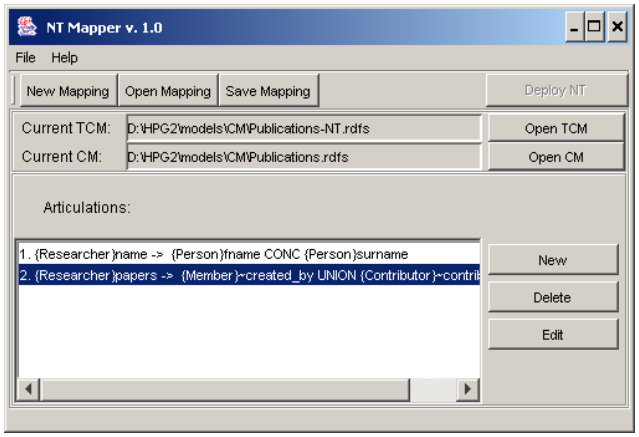


Figure 10. The main window of the mapping tool

5 Conclusion

In this paper we have shown the principles of building NT specifications that are portable over domains. These principles use existing expertise of modelling techniques and data integration. In the implementation we exploit software packages we have already developed, for instance the Hera Mediator and the EROS RDFS explorer. Although we chose a concrete (Hera) method for demonstrating the approach, we believe that the idea of mapping from a TCM to a concrete CM is rather universal. The advantage of our method compared to some other approaches lies in the possibility of precise specification of the navigation structure and the data manipulation within an NT that is subsequently automatically transformed to appropriate specification matching a concrete domain. Thus, this approach and its implementation will facilitate the reuse of navigation primitives in web engineering.

References

[1] Openrdf, the serql query language, rev. 1.1. `http://www.openrdf.org/doc/users/ch06.html`.

[2] D. Brickley and R. V. Guha. Rdf vocabulary description language 1.0: Rdf schema. *W3C Recommandation 10 February 2004*.

[3] S. Ceri, P. Fraternalli, A. Bongio, M. Brambilla, S. Comai, and M. Matera. *Designing Data-Intensive Web Applications*. Morgan Kaufmann Publishers Inc., 2003.

[4] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns*. Addison Wesley, Reading, MA, 1995.

[5] J. Gomez and C. Cachero. Oo-h method: Extending uml to model web interfaces. *Idea Group Publishing*, pages 144–173, 2003.

[6] G. Houben, F. Frasincar, P. Barna, and R. Vdovjak. Modeling user input and hypermedia dynamics in hera. In *International Conference on Web Engineering (ICWE 2004), Munich, Germany*, 2004.

[7] N. Koch, A. Kraus, and R. Hennicker. The authoring process of the uml-based web engineering approach. In *Proceedings of The First International Workshop of Web-Oriented Software Technology*, 2001.

[8] F. Mannola and E. Miller. Rdf primer. *W3C Recommandation 10 February 2004*.

[9] J. Miller J., Mukerji. Mda guide version 1.0.1. *OMG*, June 2003.

[10] O. Pastor, J. Fons, and V. Pelechano. Oows: A method to develop web applications from web-oriented conceptual models. In *Proceedings of International Workshop on Web Oriented Software Technology (IWWOST)*, 2003.

[11] G. Rossi, F. Lyardet, and D. Schwabe. Patterns for e-commerce applications. In *Proceedings of Europlop 2000*, 2000.

[12] K. Sattler, S. Conrad, and G. Saake. Interactive example-driven integration and reconciliation for accessing database federations. *Inf. Syst.*, 28(5):393–414, 2003.

[13] D. Schwabe, G. Rossi, L. Esmeraldo, and F. Lyardet. Engineering web applications for reuse. *IEEE Multimedia*, pages 2–12, Spring 2001.

[14] A. Sheth and V. Kashyap. So far (schematically) yet so near (semantically). In *Proceedings of the IFIP WG 2.6 Database Semantics Conference on Interoperable Database Systems (DS-5)*. North-Holland, 1993.

[15] R. Vdovjak, P. Barna, and G. J. Houben. Eros: A user interface for the semantic web. In *7th World Multiconference on Systemics, Cybernetics and Informatics*, 2003.

[16] R. Vdovjak, F. Frasincar, G. J. Houben, and P. Barna. Engineering semantic web information systems in hera. *Journal of Web Engineering (JWE), Rinton Press*, 2(1-2):3–26, 2002.