User Profiling Capabilities in OOWS

J. Fons[1], F.J. García[2], V. Pelechano[1], and O. Pastor[2]

[1] Department of Information Systems and Computation – Valencia University of Technology
{jjfons,pele,opastor}@dsic.upv.es
[2] Department of Computer Science – University of Salamanca
fgarcia@usal.es

Abstract. Personalisation and adaptation capabilities are significantly presented in today web applications. User profiling and user modelling are key activities to achieve the required personalised and adaptive levels in web domains. Adaptive hypermedia systems offer an adequate solution to this demand, but we argue the need to approach these issues from the beginning of the software life cycle. Object-Oriented Web-Solutions Modelling (OOWS) is a UML-based Web Engineering method to develop web applications that is strongly based on conceptual modelling techniques. OOWS has abstract primitives that allow the specification of the user profiles from the very first step of the application life cycle. This paper introduces how OOWS supports the user profiling activities that are the basis to develop personalised and adaptable web applications for specific user requirements.

Keywords. User profiling; Personalisation; Adaptable web applications; Conceptual modelling; Web Engineering.

1 Introduction

Traditional web sites consist of fairly static information. The WWW currently are formed by several millions of information pages across millions of web sites, and nowadays there are more than 500 millions of regular users from all walks of life with varying degrees of computer expertise, a wide range of interests and preferences, and a multimodal access representations or differing capacities to make use of them. Offering the right information for a particular web site user is one of the most interesting challenges of the Web Engineering.

Adaptive Hypermedia [2], content personalisation [16] and user modelling [11, 20] areas have presented a number of potential solutions to reduce the information overload by automatically learning about likes and dislikes of individual users in order to personalise the information retrieval, the services behaviour or the navigation.

User profiling and user modelling are key activities to achieve the required personalised and adaptive levels in web applications.

User profiling is the process of identifying and categorising the user audience of a Web site, gathering statistics on them. People from the different profiles, which are identified in a concrete web system, may perform similar tasks on the site; however, the exact goals may change slightly, and their concerns will differ. So, it is important to understand the differing needs of these groups of users, otherwise the site may succeed with one group, but fail with another.

All the information about of each individual user of the web site (interests, navigation, environment…) is stored in its user profile. We call user model to the explicit representation of the properties of a particular user [9]. The goal of user modelling is to create systems that are adaptive to an individual's needs, abilities, and preferences. The user model should be central to the design process and substantially influences the basic functionality of web applications [18], because the user model is the source for personalising the content and navigation.

For this reason, we claim that the personalisation and adaptive issues in web applications should be approached from the beginning of the software process in the Web Engineering methods, i.e. from the requirements elicitation and especially in conceptual modelling.

Object-Oriented Web-Solutions Modelling (OOWS) [13, 14] is a UML-compliant [12] Web Engineering method (developed by the OO-M Group at Valencia University of Technology) to design web applications that is strongly based in the conceptual modelling techniques.

The aim of this paper is presenting how OOWS supports user profiling through a set of conceptual models and abstract primitives, this way, this method deals with the design of personalised and adaptable web applications.

The rest of the paper is organised as follows. Section 2 gives a general overview of the OOWS method. Section 3 focuses in the OOWS primitives that allow the user profiling. Section 4 presents a comparison with related works. Finally, section 5 provides remarks and further work.

2 OOWS Overview

OOWS is an extension of the OO-Method approach [15] that allows web applications development. OOWS enriches OO-Method with a navigation and presentation models to capture navigational and user interface aspects.

The software production process comprises two major steps: *specifying the system* and *developing the solution*. At first, a full specification of the user requirements is built in the *specifying the system* step. A strategy oriented towards generating the software components that constitute the solution (the final software product) is defined in the second step.

In order to make easier the explanation of the primitives that contribute to the user profiling modelling in the next section, we are going to present the specification step in a deeper way now.

2.1 Specifying the System

This step includes requirements elicitation based on use cases and scenarios (see detailed information in [8]) and system conceptual modelling activities. When dealing with the conceptual modelling stage, the abstractions derived from the problem space are specified in terms of their classes, structure, behaviour and functionality.

OOWS uses UML-compliant diagrams to represent the required conceptual information in five models: the *Object Model* that is represented by means of a class diagram; the *Dynamic Model* that is used to specify valid object lives and interobjectual interaction; the *Functional Model* that captures the semantic associated to the changes of state of the objects motivated by the services occurrences; the *Navigational Model* that specifies the navigation semantics associated to the system users using a UML notation too; and the *Presentation Model* that uses presentation patterns to specify the format of the output presentation.

The navigational Model is built using five main basic abstraction primitives: Navigational Map, Navigational Context, Navigational Link, Navigational Class, and Navigational Relationship.

The Navigational Model is essentially composed of a set of Navigational Maps that represents the global view of the web system for every potential end-user. It is represented by a directed graph in which the nodes are the Navigational Contexts and the arcs are the Navigational Links defining valid navigation paths.

A Navigational Context permits the definition of the content of user interactions units. It is composed by Navigational Classes and Navigational Relationships.

A Navigational Class defines a view of an object model class, and the Navigational Classes are connected by Navigational Relationships that describe valid navigation paths over object relationships.

There are Navigational Contexts of two kinds: *exploration contexts* that can be reached at any moment, independently of the current context; and *sequence contexts* that can only be reached by following a predefined sequence of Navigational Links.

3 User Profiling in OOWS

OOWS supports user profiling in the conceptual modelling phase by the different models that have been presented [1]. The overall models have to express the relevant information that allows creating a user profile. We have to act in three levels [17].

- *Content level*: which information is available for every kind of user, and also which information has to know the system about the different kinds of users.
- *Navigation level*: which links are available for every kind of user.
- *Presentation level*: how the information is shown to every kind of user.

In order to make a conceptual model of a web application, taking the user profiling capabilities, the steps that should be done are:

1. Find the user types that represent all the end-users of the web application.
2. Express every possibility of changing in the user type.
3. Model a view of the system for each user type, including the content, the navigation and the presentation levels.

3.1 Modelling User Types

The first activity in user profiling is to know the user audience of the system. In order to perform this activity we should gather all internal information on the users, including user feedback, survey results, customer support information, market research and so on.

All the information gathered should be analysed before creating realistic profiles for each segment of the user audience. The result of this analysis is a diagram representing all the user types (so called agents in OO-Method and in OOWS too) for the web applications.

Each user type represents a group of end-users that share all the properties and characteristics in the web applications at content, navigation and presentation levels.

It would be very interesting that the identified agents would be able to relate themselves by generalisation/specialisation relationships. This situation makes easy modelling the views of system for each user type presented in the system, because all the things modelled for an agent will be truth for a more specific one applying the substitution principle.

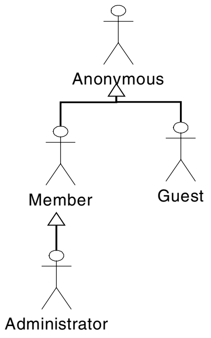

Fig. 1. Agent diagram

For example, the Figure 1 shows an agent hierarchy that expresses a typical situation in web applications. There exists an `Anonymous` agent that represents the most generic user type in the system, which is characterised because it has a very limited access to the system. Two new subtypes appear as direct descendants of the `Anonymous` agent: the `Member` agent and the `Guest` agent, which have a more specific behaviour in the system, but every end-user that belongs to one of these categories, can perform every `Anonymous` agent's behaviour. Finally, a new user type appears, the `Administrator` agent, which is a specialisation of the `Member` agent.

The agent diagram (see Figure 1), uses a UML-like notation where the agent is represented by the actor stereotype (with a stick man icon), and the relationships among agents is represented by the generalisation/specialisation connection.

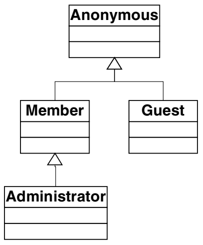

Fig. 2. Piece of the class diagram related to users

In order to register the information for each individual profile, this agent diagram is directly mapped to a class taxonomy in the class diagram of the OOWS approach. In this way, it is possible to provide more detailed information for a specific user that can be relevant to personalise the system. Fig. 2 shows the piece of the class diagram related to the users for the example illustrated in Figure 1. This information plays a very important role to achieve individual personalisation and adaptation.

3.2 Changing User Types in Navigation Models

Frequently in web applications an end-user can change its system view. For example, an end-user enters at the system as an anonymous user, but when the user needs more privileges or a specialised access, it authenticates itself and gains the proper level at the web system. To represent systems that define an intrinsic way of promoting the interactive user, OOWS allows specifying a UML State Transition Diagram (STD) attached to the Navigational Model[1]. This STD help us to model valid interactive user type changes during navigation. The STD states represent the possible types of users in the system, while the transitions express valid user type changes[2].

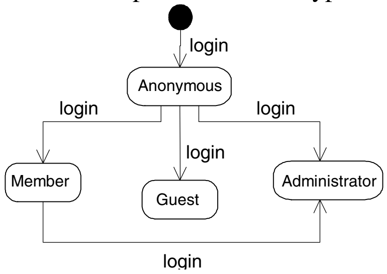

Fig. 3. Example of state chart expressing valid user type changes

Figure 3 shows an example of every possible user type change in a particular web system. In this example, by default, a user must be authenticated as an `Anonymous`[3]. Then, he can only *login as* a `Member`, `Guest` or `Administrator` user types, and being `Member` he can authenticate as `Administrator`.

3.3 Designing Personalised Views

According to [17], we only build high quality personalised and adaptable web applications if we design those applications with flexibility and extension in mind from the beginning.

Till now, we have identified the type of users that can interact with a web system, but we have to design for each user type its content, navigation and presentation properties. To do that, we specify the navigational and presentation model.

[1] If this STD is not specified, users can change their interactive type with no restrictions.
[2] Every change of user needs an authentication (login).
[3] A login and a password are not necessary.

The *navigational model* is made up of a navigational map for each identified agent. A *navigational map* represents the global view of the system for this group of end-users. This map is composed by a set of navigational contexts (depicted as UML packages stereotyped with the «context» reserved word) related by navigational links (depicted as arrows) defining the navigational (structure) personalisation for this type of user.

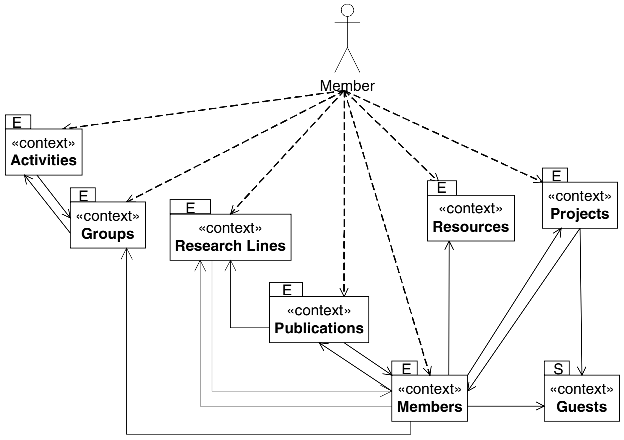

Fig. 4. Example of navigation map

For example, Figure 4 shows the navigational map for an anonymous agent of a web system. This map contains the navigational contexts that can be accessed by users of this kind and the valid navigational paths (sequence of contexts) than an anonymous user can follow. In the same way, a navigational map is built for each type of user, personalising its navigational capabilities.

Now, the navigation contexts are defined according to the user type that leads the navigation map. Each navigation context includes a configuration of navigation classes and navigation relationships that represents a view of a subset of the object model. This view expresses the attributes and the operations that are allowed for this user type, providing personalised access for this type of user.

Figure 5 shows the `Members` context from the point of view of an anonymous user. In the current case appears four navigation classes: the `Member` class, which is the *manager class* of this context (the main class of this context), and the `WorkOn`, `Entity` and `RGroup` classes, which are the *complementary classes* (contributing to give additional information).

Besides, in this example three navigational relationships appear. The first two (depicted with dashed arrows) are *contextual dependency relationships,* causing an object retrieval of the related instances (using a structural relationship, i.e. association, aggregation, composition, specialisation/generalisation). In the example, they relate the `Member` class with the `WorkOn` class and that `WorkOn` with the `Entity`. The last one is a *context relationship* (also a unidirectional binary relationship, depicted with a solid arrow) that relates the `Member` class with the `RGroup` class. Relationships of this kind act as contextual dependency relationships and they also define a navigation capability to a *[target context]* in the map (in this example, to the `[RGroup]` context).

Both kinds of relationships are defined on existing classes and class relationships from the class diagram, assuring full compatibility between the models of the conceptual modelling.

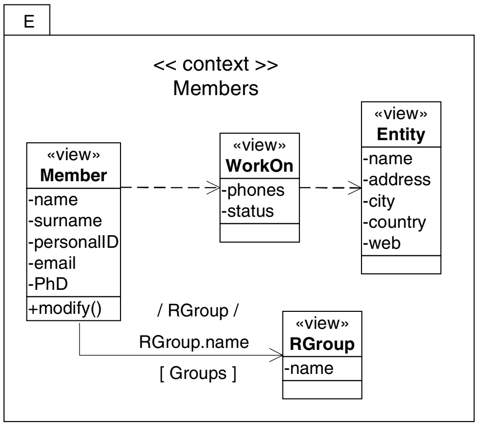

Fig. 5. Example of navigation context

Navigational contexts express the content personalisation for a type of user. Finally, for the presentation information we have to define the *presentation model* of the OOWS approach. This model allows specifying abstract presentation requirements of a web application related to each navigational context. Figure 6 shows an example of a navigational context with presentation information for the anonymous type of user.

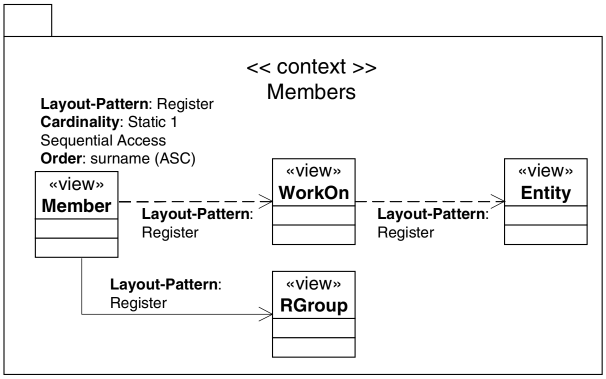
<< context >>
Members
Layout-Pattern: Register
Cardinality: Static 1
Sequential Access
Order: surname (ASC)
«view»
Member
«view»
WorkOn
«view»
Entity
Layout-Pattern: Register
Layout-Pattern: Register
Layout-Pattern: Register
«view»
RGroup

Fig. 6. Presentation information for a specific navigation context

4 Related Works

Nowadays, the approaches that address some kind of personalisation vary widely [16]: from single page generation strategies to complex content-based prediction systems, data mining, machine-learning algorithms... Most of these approaches consider personalisation and adaptation issues from an implementation perspective.

However, we consider the customisation issues in general should be treated from the conceptual modelling tasks inside of well-defined software engineering methods. Now, we are going to compare the personalisation issues presented in OOWS with other interesting conceptual modelling based Web Engineering approaches (WSDM [5], OOHDM [19], WebML [3], OO-H Method [7] and UWE [10]).

The Web Site Design Method (WSDM) is a user-driven approach that comprises three main phases: user modelling, conceptual design and implementation design. In the user modelling stage, the potential users of web site are identified and classified, this way for each potential user profile are systematically gathered and a class diagram of user profiles is built. The navigational model consists of a set of navigation tracks, each one specified for a particular user perspective. Every navigation track comprises three layers: context, navigation and information; this approach achieves very hierarchical web application compared to the flexible structured derived from the OOWS navigation model. On the other hand, WSDM defines its own graphical notation for the objects of the navigation model, and this fact is a very important drawback compared to OOWS, for example, which is UML-compliant. Lastly, the WSDM implementation design step looks for creating a consistent look and feel for the conceptual model, but this step is not very rich in recommendations, in this sense OOWS has a rich pattern-based presentation model and a generative approach for the automatic final product generation.

Object-Oriented Hypermedia Design Method (OOHDM) presents four activities: conceptual modelling, navigation design, abstract interface design and implementation. The notation of OOHDM is close but not completely compliant to UML. This method has been extended to specify personalised web systems by defining different scenarios according to user profiles or preferences [17]. One of the most representative differences with OOWS is that OOHDM does not allow objects to offer services to the users of application.

WebML is a specification language that gives a good basis for specifying different aspects of a web application such as content structure, page composition, navigation, presentation and personalisation. This language has its own graphical notation (not UML-compliant) that includes an explicit notion of group and user, where group describe sets of users with common characteristics (agent in OOWS), while users denote individuals. WebML is data-oriented while OOWS is object-oriented. This difference could be important when applications grew in size and complexity.

OO-H Method and OOWS share the same conceptual model, OO-Method [15], and also both have the goal of allowing an automatic web application generation. The main difference between OO-H Method and OOWS is that OOWS is a full OO-Method extension that includes the functional specification completely, meanwhile OO-H only specifies the interface. Another difference appears in the navigation model, more specifically in the concept of node. OO-H Method associates a different navigation access diagram with each user-type, but the nodes (navigation classes) are limited to present information of only one class, whereas nodes in OOWS (navigational contexts) can work with views of several classes, showing a more appropriate information for the user in every moment. OO-H Method is now being extended with an adaptation model to specify adaptive and proactive personalised web applications [6].

The UML-based Web Engineering (UWE) is a key reference of an engineered approach to the development of web applications with a special focus on personalisation and adaptation issues. It is based on the so called Munich Reference Model that describes the navigation mechanism of a web application. This model presents a three-layer structure (run-time layer, storage layer, within-component layer) that includes the needed functionality to support user modelling and adaptation aspects. UWE is a highly relevant reference in order to extend OOWS with the adaptive model, but also OOWS can contribute with a solid full life-cycle engineering process with automatic code generation capabilities.

5 Conclusions and Further Work

In this paper, we have presented how OOWS, a web application production process, deals with the user profiling activities in order to achieve personalisation and adaptation issues in web systems.

We have argued that our proposal allows the specification of the user profiles from the conceptual modelling at the beginning of the application life cycle. This way, our approach deals with other important research works that defend a more engineered treatment of the customisation issues in comparison to the most extended approach that considers the user profiling from an implementation point of view.

In order to achieve this goal, OOWS presents a set of UML-compliant abstract primitives that appear in the different models of its specification systems phase. Essentially, the user profiling primitives allow modelling the different user types that represent the allowed groups in the web systems, modelling the possible user type changes and modelling the content, presentation and navigation capabilities for each group. Limiting to the notation proposed by the UML instead of introducing a new notation has the advantage of using a well-known standard and that UML is supported by many case tools. UML is extended to model the navigation and the presentation according to the UML extension mechanisms.

With these arguments we can say OOWS supports the specification and design of personalised and adaptable systems, but actually this method fails in individual personalisation support. For this reason, according to [4], OOWS deals with the so called adaptable hypermedia systems but does not support the so called adaptive hypermedia systems.

Further work is mainly directed to define a new model in the conceptual modelling phase that gives an adequate support of adaptive properties but assuring a full compatibility between this new adaptation model and the other models used in the process of conceptual modelling in OOWS.

Acknowledgements. Research supported by a CICYT Project (ref. TIC2000-1673-C06-05), a Government of Castile and León Project (ref. SA017/02) and a MCYT Project (ref. TIC2001-3530-C02-01).

References

[1] Abrahão S., Fons J., González M., Pastor O. Conceptual Modeling of Personalized Web Applications. In *2nd International Conference on Adaptive Hypermedia and Adaptive Web Based Systems, AH2002*, Lecture Notes in Computer Science. LNCS 2347. Springer Verlag (2002) 358–362.

[2] Brusilovsky, P.: Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, Vol. 11 (2001) 87–110.

[3] Ceri, S., Fraternali, P., Paraboschi, S.: "Web Modeling Language", (WebML): A Modeling Language for Designing Web Sites. In *Proceedings of the 9th International World Wide Web Conference WWW9*, Elsevier (2000) 137–157.

[4] De Bra, P.: Design Issues in Adaptive Web-Site Development. *Proceedings of the 2nd Workshop on Adaptive Systems and User Modeling on the WWW* (1999).

[5] De Troyer, O. y Leune, C.: WSDN: A User-Centered Design Method for Web Sites. In *Proceedings of the 7th International World Wide Web Conference WWW6*. Elsevier (1997) 85–94.

[6] Garrigós, I., Cachero, C., Gómez, J.: Modelado Conceptual de Aplicaciones Adaptivas y Proactivas en OO-H. In *Proceedings of the 2nd Taller de Ingeniería Web – Webe02*. El Escorial – Madrid (Spain). http://www.dlsi.ua.es/webe02/programa.htm (2002).

[7] Gómez, J., Cachero, C., Pastor, O.: Extending a Conceptual Modeling Approach to Web Application Design. In *Proceedings of Conference on Advanced Information Systems Engineering (CAiSE)*. Lecture Notes in Computer Science. LNCS 1789. Springer Verlag (2000) 79–93.

[8] Insfrán E., Pastor O., Wieringa R.: Requirements Engineering-Based Conceptual Modelling. *Requirements Engineering* Vol7, N. 2 (2002) 61–72.

[9] Jameson, A., Paris, C., Tasso, C.: Preface. In A. Jameson, C. Paris, C. Tasso (Eds.), *User Modeling: Proceedings of the Sixth International Conference UM'97*. Springer-Verlag. (1997).

[10] Koch, N.: Software Engineering for Adaptive Hypermedia Applications. PhD. Thesis (2000).

[11] McTear, M.: Special Issue on User Modeling. *Artificial Intelligence Review Journal*, Vol. 7 (1993).

[12] OMG: OMG Unified Modeling Language Specification. Version 1.4. Object Management Group Inc. http://www.omg.org/uml (2001).

[13] Pastor, O., Abrahão, S. M., Fons, J. J.: Building E-Commerce Applications from Object-Oriented Conceptual Models. *SIGecom Exchanges, Newsletter of the ACM Special Interest Group on E-commerce,* Vol. 2, N. 2 (2001) 28–36.

[14] Pastor, O., Abrahão, S., Fons, J. J.: An Object-Oriented Approach to Automate Web Applications Development. In K. Bauknecht, S. K. Madria, G. Pernul (Eds.), *Electronic Commerce and Web Technologies, Second International Conference, EC-Web 2001 Munich, Germany, September 4–6, 2001, Proceedings*. Lecture Notes in Computer Science. LNCS 2115. Springer Verlag, (2001) 16–28.

[15] Pastor, O., Gómez, J., Insfrán, E., Pelechano, V.: The OO-Method Approach for Information Systems Modelling: From Object-Oriented Conceptual Modeling to Automated Programming. *Information Systems*, Vol. 26, N. 7 (2001) 507–534.

[16] Riecken, D. (Guest Ed.): Special Issue on Personalization. *Communications of the ACM*, Vol. 43, N. 8 (2000).

[17] Rossi, G., Schwabe, D., Guimarães, R. M.: Designing Personalized Web Applications. In *Proceedings of the Tenth International World Wide Web Conference WWW10*, Hong Kong, (2001).

[18] Scharl, A.: A Classification of Web Adaptivity: Tailoring Content and Navigational Systems of Advanced Web Applications. In S. Murugesan, Y. Deshpande (Eds.), *Web Engineering. Managing Diversity and Complexity of Web Application Development*. Lecture Notes in Computer Science. LNCS 2016. Springer Verlag, (2001) 156-169.
[19] Schwabe, D., Rossi, G.: The Object-Oriented Hypermedia Design Model. *Communications of the ACM*, Vol. 38, N. 8 (1995) 45–46.
[20] User Modeling Home Page. http://www.um.org/. (2002).