Modelling Dynamic Personalization in Web Applications *

Irene Garrigós[1], Jaime Gómez[1], and Cristina Cachero[1]

IWAD Group
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante. SPAIN
{igarrigos,jgomez,ccachero}@dlsi.ua.es
http://www.dlsi.ua.es/iwad/

Abstract Conceptual Modelling approaches for the web need extensions to specify dynamic personalization properties in order to design more powerful web applications. Current approaches provide techniques to support dynamic personalization, usually focused on implementation details. This article presents an extension of the OO-H conceptual modelling approach to address the particulars associated with the design and specification of dynamic personalization. We describe how conventional navigation and presentation diagrams are influenced by personalization properties. In order to model the variable part of the interface logic OO-H has a personalization architecture that leans on a rule engine. Rules are defined based on a *User Model* and a *Reference Model*. OO-H provides two type of rules: acquisition rules, in which the needed information is collected, and personalization rules that are applied to the navigation and presentation level, and reflected in their corresponding conceptual views. In this way, the interface logic of a web application is viewed as a composition of a stable and a variable part, where the variable part (expressed in XML) is interpreted at execution time. The main benefit is that this specification can be modified without recompile the rest of the application modules.

1 Introduction

Current Web Engineering approaches [6] help designers to make easier the understanding, development, evolution and maintenance of web applications. These methods are based on new constructors and hypermedial views [9, 12, 2, 13] that broach the problem of navigation/presentation of the user across the information space.

However, the approaches that address some kind of personalization vary widely. In this context, the main problem is the lack of conceptual modelling constructs with dynamic personalization support. We argue that new techniques to extend metamodels with personalization aspects are needed. Also conceptual models must be preserved if we want to take advantage of previous modelling efforts. Furthermore, we need to address personalization by means of an externalization process where personalization rules can be interpreted at execution time. In this way, we can provide a flexible personalization treatment that is technology-independent.

This article presents how the *Object Oriented Hypermedia Method (OO-H)* [8, 7, 1] is extended to support dynamic personalization. We describe how conventional navigation and presentation diagrams are influenced by personalization properties. These properties are captured in the form of external files written in XML, that represent the rules. These rules, that form the variable part of the interface logic, will be treated at execution time by an engine included in the execution architecture. The support for this architecture is achieved by two models:

- a reference model, that also registers the user activity in the system.
- a user model that collects the needed information to personalize.

* This paper has been partially supported by the Spain Ministry of Science and Technology, project number TIC2001-3530-C02-02.

In this way, the interface logic of a web application is viewed as a composition of a stable and a variable part, where the variable part (expressed in XML) is interpreted at execution time.

The remainder of the article is structured as follows: section 2 presents related work in the field of dynamic personalization. Section 3 shows the elements that support the personalization in OO-H. Namely, subsection 3.1 presents the underlying execution architecture of the OO-H web applications. Subsection 3.2 describes how the knowledge that the system has about the user is captured by means of a user model. Subsection 3.3 presents how the modelling strategy of personalization is defined by means of a set of information structures expressed in a reference model. Section 4 presents the concept of association rules to support the specification of dynamic personalization. Section 5 describes, by means of a comprehensive example, how the personalization of content, navigation and presentation, is achieved in a separated way. Section 6 shows how these rules are made effective in the navigation and presentation models of OO-H. Finally, section 7 presents the conclusions and further work.

2 Related Work

With respect to Web Engineering approaches, some of them provide support for personalization depending on roles, like OOHDM [12] or WSDM [13].

In OOHDM the abstract interface model is the result of the specification of the interface objects the user will perceive. OOHDM uses *Abstract Data Views*(ADVs) to model the static aspects of the user interface [10] while dynamic aspects of the user interface are modelled with a technique based on Statecharts [3]. The modelling and design in a conceptual frame allow a better understanding of the used mechanisms and the discovery of common features that allow to reuse patterns, components, algorithms or even subsystems [12]. In the WSDM approach [13] different perspectives are defined for the user classes; these are different ways in which different types of users look at the same information and navigate through the information.

Other approaches provide frames that complement the conceptual model and support adaptation, adaptivity or even proactivity features, as W3I3 [2] or UWE [9].

W3I3 [2] includes mechanisms to externalize the policies and to support their change once the application has been deployed. However, these mechanisms are implemented as database triggers and are focused on the design of data-intensive web applications.

UWE [9] is centered in the specification of adaptive applications. It insists on personalization features, such as the definition of a user model or a set of adaptive navigation features that depend on preferences, knowledge or tasks that the user has to execute. From our point of view this is the most complete proposal. It provides a formal theoretical framework for dynamic personalization. However, the main problem is the inflexibility of the UWE conceptual models. We mean that any change the designer wants to introduce must be supported in the UWE framework.

Finally, there is a big number of commercial tools (e.g. ILog JRules, LikeMinds, WebSphere, Rainbow..) that make easier the use of the personalization techniques and strategies and give support to many personalized web applications. These tools are oriented to the implementation of personalization strategies. The main problem is the low abstraction level that causes reuse problems and a difficult maintenance and scalability of the resultant personalized applications.

Next, we are going to present how OO-H overcomes some of these limitations.

3 Personalization Support in OO-H

3.1 Personalization Architecture

The architecture of OO-H has been extended to support dynamic personalization. Fig. 1 shows the new architecture. More specifically, personalization properties are captured at navigation/presentation level and are reflected in their corresponding conceptual models (NAD, APD) by means of a set of association rules. In this way the design and the generation of the navigation logic is specified in two parts: a stable part, that is independent from the properties of personalization, and a variable part, that supports the treatment of these rules. Finally, a rule engine provides the context to interpret the generated rules at execution time.

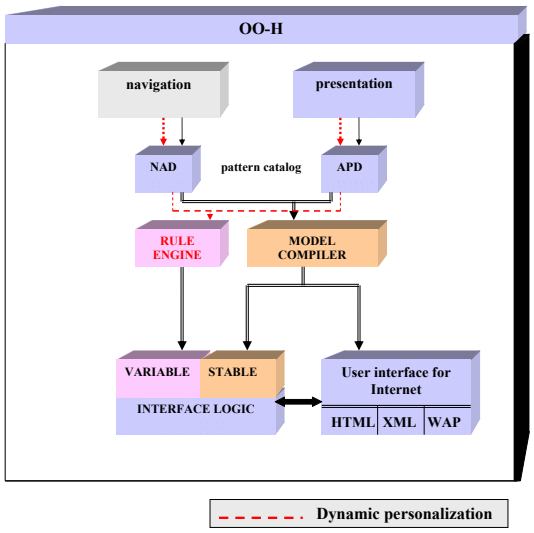

Figure1. *OO-H Architecture with dynamic personalization*

Modelling the variable and stable parts of the application in a separated way from the first phases of the software life cycle, facilitates the treatment of the dynamic nature of personalization features of web applications, as well as the reusability and performance of changes. The justification of this statement is similar to the reasons argued to separate business policies from object oriented applications [11]. The support for this architecture is achieved by a *Reference Model* that allows to capture the relevant properties of personalization and is presented in section 3.3. Moreover, a *User Model* must be defined to support the personalization requirements and that will be presented in the following section.

3.2 The User Model

The user model is an important part of an adaptive hypermedia system since the information kept in the user model allows to modify or adjust information (adaptive content), provide the user with navigation support (adaptive navigation) or individualize layout (adaptive presentation). It is defined as the representation of the system's beliefs or knowledge that the system has about the user. A user model is constituted by descriptions of what is considered relevant about the actual knowledge and/or aptitudes of a user, providing information for the system environment to adapt itself to the individual user [9].

In OO-H the user model is captured by means of a Class Diagram that complements the Conceptual model. It captures information about the features that the system believes the user has. This

features can be preferences or interests, general knowledge, experience, etc. A user model does not need to be completely accurate, and in fact it is usually restricted to rough approximations. However, even an incomplete user model can be useful [9]. The information that should contain this model depends on the personalization requirements that we want to support.

In OO-H the user model is centered on the concepts of user and role, the same that in another hypermedial approaches [2]. For the provision of personalized views to the user, OO-H provides a basic user model. This user model is constructed around a class named *User*, that provides the information and behaviour that must be inherited by every <<*actor*>> of the system. A user may not have a role associated, in this case s/he would be treated as an *anonymous user*. Moreover, a user can only have associated a role at the same time. This model can be enriched to support the desired personalization policy adding attributes, methods or links from the class User tothe rest of the classes of the domain or the *OO-H Reference Model*, which is presented in the next section. The construction of a user model will be seen in the section 5.

The modelling strategy of personalization must be defined depending on a set of information structures. This information is stored in OO-H in a repository that contains the initial set of basic elements of information on which the desired personalization policy can be established. This is what is called the reference model.

3.3 The Reference Model

The main benefit the use of a reference model provides is that the designer can include and connect this framework with any model of the OO-H interface to which s/he wants to endow with the ability of personalization. From there, OO-H allows the extension of this repository with the particular features that the application requires, therefore, not being a closed frame. A first version of this repository was presented in [5]. The current OO-H reference model (see Fig 2) structures the modelling of personalization in OO-H in three parts: user profiles, context information and association rules. For the purpose of this paper we must pay more attention to the part B of Fig. 2, it is, to the association rules. A new type of rule has been added to this repository: the presentation rule (see Fig. 2).

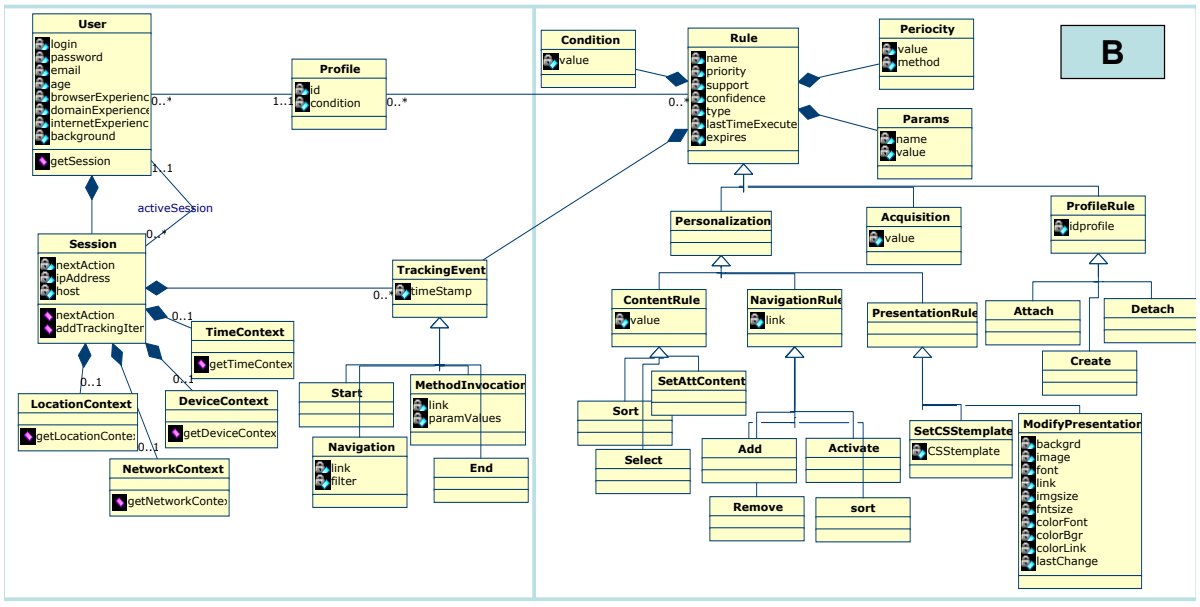

Figure2. *Personalization Framework*

The **Association Rules** are presented in depth in the following section. As we will show, this type of rules allow to capture the personalization properties embedded directly in the models of OO-H.

4 Specification of the Personalization in OO-H

OO-H incorporates an external mechanism to model the personalization of the adaptive and proactive applications in form of association rules. An advantage of using these rules is that they are defined in external files and can be modified in an independent way from the rest of the application.

The rules have been divided in three types:

- **Acquisition Rules**: in which the needed information for personalization is collected.
- **Personalization Rules**: support the action of personalization. As we have seen in the Fig. 2, this type of rules are also categorized, based on what we want to personalize, in:
  - **Content Rules**: using these rules we can select *(Select)*, sort *(Sort)* or modify *(SetAttContent)* the content information of the application.
  - **Navigation Rules**: the rule affects to the navigation mode. With this type of rule we can *Add, Remove, Activate* or *Sort* any links in the user navigation.
  - **Presentation Rules**: the rule affects to presentation aspects. The possible actions that can be associated to this type of rule are: *SetCSSTemplate or ModifyPresentation*, according to wether we want to use a predefined template of presentation or we want to modify a specific value. We have a set of CSS templates, that can be associated to specific cases (as for disabled persons) captured in the *Reference Model* and a *Presentation Class* with presentation features (see section 3.3). Moreover, in the *User Model* we can complete this Presentation Class with the features designed for a specific conceptual model (see Fig. 3). The rules permit the selection of a predefined template or the modification of specific features of the Presentation Class. We can see an example in section 5.
- **Profile Rules**: these are profiles management rules. With these rules we can associate *(Attach)* or disassociate *(Detach)* specific behaviours to user profiles, or we can create a new profile *(Create)*.

OO-H supports the specification of these rules by means of an XML schema [14]. Is important to note that the execution architecture of the generated applications from OO-H models allows to modify and reprocess this schema without recompiling the rest of modules. In the schema the different XML elements correspond to classes/attributes of the frame presented in Fig. 2.

All the rules are ECA (Event-Condition-Action) rules [4]. Moreover, in the *Acquisition rules* a *Periocity* of the rule can be established. It specifies the interval in which, in an automatic way, the application has to check that rule. In the context of the periocity tag, OO-H defines a special value, *"ooh:always"*, that indicates that the accomplishment of the activation conditions must be checked in a continuous way. The action that may implement a rule varies depending on the type of the rule, like we have seen.

In the next section we will present an example of the three types of dynamic personalization that can be modelled in OO-H: Personalization of Content, Navigation and Presentation. These rules will be applied on a video club in Internet that manages clients, movies and rentals.

5 Modelling example

The *User Model* and the *Reference Model*, besides the *Association Rules*, let us to personalize the content, the navigation and the presentation of an application.
In the context of the mentioned system (video club in Internet) we are going to consider the following requirements:

- When a movie is rented, show recommendations of movies of the same category *(Content Personalization)*.

– Show a link to see the movie trailer in the case that the user has a proper plug-in *(Navigation Personalization).*
– In the case of a vision deficiency make bigger the font *(Presentation Personalization).*

The first step to support these requirements is to define the *User Model*. It is a Class Diagram that complements the Conceptual model, as we said in section 3.2. A user model is constituted by descriptions of what is considered relevant about the actual knowledge and/or aptitudes of a user, providing information for the system environment to adapt itself to the individual user [9]. In this case one possible user model can be seen in the Fig. 3.

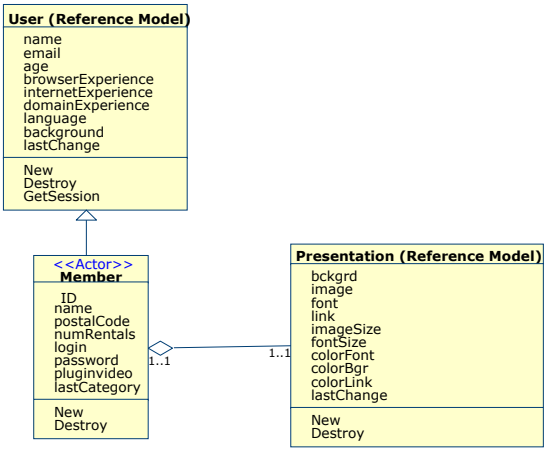

Figure3. *User Model*

This user model results from the connection of the *User* class, from the reference model, with the existing class *Member*, from the conceptual model, and treats it as a role. It is also added another class from the reference model, the *Presentation* Class, that stores the presentation preferences of each individual user.

Moreover, to specify the personalization we have to define the way to support each one of these requirements. Let's see them one by one. In all of the requirements we have to distinguish between the acquisition and personalization of the corresponding data. The support of each of the requirements is detailed in three phases:

– *Modelling and Gathering process*, where it is shown how the requirement is introduced in the NAD/CLD diagram.
– *XML specification,* where it is shown an XML specification that supports the requirement.
– *Final Representation,* where it is shown the storyboard of the corresponding requirement.

5.1 Personalization of Content

The content personalization requirement is the following:
When a movie is rented, show recommendations of movies of the same category.

Acquisition Rule (1a):

MODELLING AND GATHERING PROCESS

Figure 4 shows a portion of a NAD diagram in which the rule that models this requirement is introduced. When the service link 'rent' is double clicked, the dialog box shown in Fig. 4 appears to introduce the needed parameters to model the acquisition rule. These parameters are:

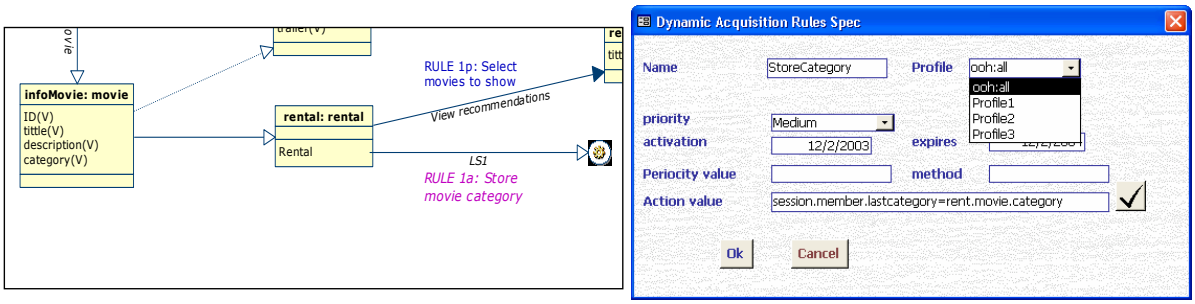

Figure4. *Acquisition: rule 1a*

- *Name of the rule*
- *Profile:* it can be selected from all the existing profiles that have been created, or the special value "ooh:all" that indicates that the rule will be applied to all profiles.
- *Priority of the rule:* it can take the following values depending on the priority of execution of the rule : *low, medium, high.*
- *Date of activation of the rule*
- *Date in which the rule expires*
- *Periocity value:* the periocity in which the rule is executed if is the case of a proactive rule.
- *Method* that is used to acquire the information (if it is the case)
- *Action value:* the action that will be performed when the premises are accomplished. This value is an OCL expression that can be validated.

XML SPECIFICATION

The result of this acquisition process is stored as an XML specification that will be interpreted by a rule engine at execution time.

```
<TPersonalization>
    ...
    <profile name="ooh:all" condition="">
        <rule type="acquisition" name="StoreCategory" support="20" confidence="100" priority="Medium"
            activation="12/02/03" expires="12/02/04" lastTimeExecuted="23/09/03">
            <event type="MethodInvocation" link="Rent"/>
            <action value="session.member.lastcategory=rent.movie.category" />
        </rule>
    </profile>
    ...
</TPersonalization>
```

This specification describes how the rule affects all profiles and has medium priority. The information that we need to acquire is the category of the last rented movie. This type of acquisition is implicit and is stored when the member rents a movie in the attribute of the User Model "session.member.lastcategory".

Personalization Rule (1p):

MODELLING AND GATHERING PROCESS

Figure 5 shows the part of the NAD diagram in which the rule that models this requirement is introduced. When the link View Recommendations is activated the rule is triggered. When this link is double clicked, the dialog box shown in Fig. 5 appears to introduce the needed parameters to model the personalization rule.

These parameters are:

- *Name of the rule*

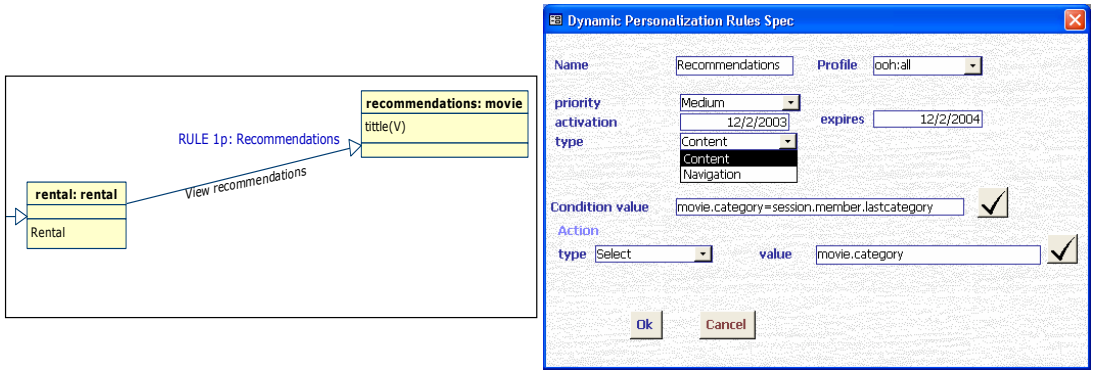

Figure5. *Personalization: rule 1p*

- *Profile:* it can be selected from all the existing profiles that have been created, or the special value "ooh:all" that indicates that the rule will be applied to all profiles.
- *Priority of the rule:* it can take the following values depending on the priority of execution of the rule : *low, medium, high.*
- *Date of activation of the rule*
- *Date in which the rule expires*
- *Type of the rule:* we can model a content or a navigation rule, in this case, the type of the rule is *content.*
- *Condition value:* This parameter is an OCL expression which has to be accomplished to trigger the rule. It can be validated by means of an OCL compiler.
- *Action type:* This tag can take as values the actions that a content rule can take: *Select, Sort, setAttContent.* In this case the chosen value is *Select.*
- *Action value:* The action that will be performed when the premises are accomplished. This value is an OCL expression that can be validated.

XML SPECIFICATION

In this case, we have content personalization, because we have to show or hide a specific information. We have as parameter the category of the last movie rented, that was obtained by the acquisition rule. The personalization event indicates that the link "View Recommendations" must be active. If these conditions are accomplished, the action will be executed: the movies with the imposed conditions are shown.

```
<TPersonalization>
    ...
    <profile name="ooh:all" condition="">
        <rule type="personalization:content" name="Recommendations" support="20" confidence="100" priority="Medium"
            activation="12/02/03" expires="12/02/04" lastTimeExecuted="23/09/03">
            <params>
            <param name="catmovie" value="session.member.lastcategory"/>
            </params>
            <event type="Navigation" link="View Recommendations"/>
            <condition value="movie.category=catmovie"/>
            <action type="Select" value="movie.category"/>
        </rule>
    </profile>
    ...
</TPersonalization>
```

FINAL REPRESENTATION

Figure 6 shows the the storyboard of this personalization rule. Once a movie is rented the list of recommendations is showed to the user.

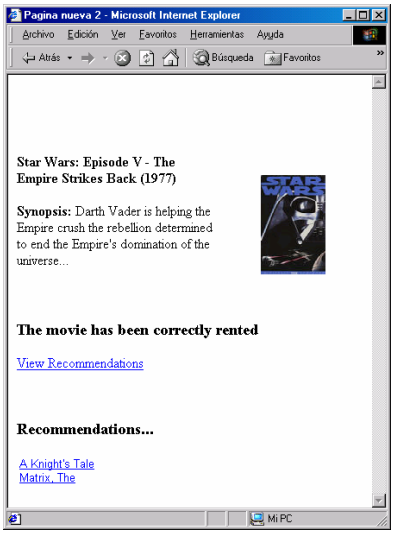

Figure6. *Final Representation: rule 1p*

5.2 Personalization of Navigation

The requirement of the Personalization of Navigation is the next one:
Show a link to see a movie trailer in the case that the user has the properly plug-in.

Acquisition Rule (2a):

MODELLING AND GATHERING PROCESS

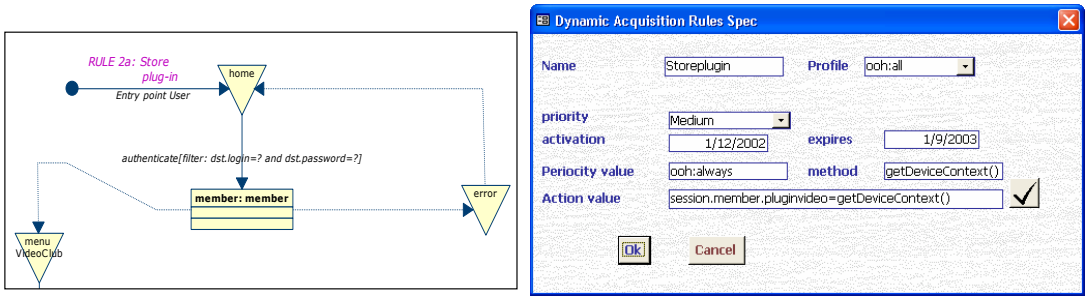

Figure7. *Acquisition: rule 2a*

When the entry point link is double clicked, the box dialog shown in Fig. 7 is showed to introduce the needed parameters to model the acquisition rule. These parameters are the same that in the rule 1a, and use the getDeviceContext(), property provided in the Reference Model to capture information about the video plug-in the user has.

XML SPECIFICATION

In this case the XML specification stores information related to a proactive acquisition. This rule is applied to all profiles and has medium priority. The periocity of execution takes the default value "ooh:always" that indicates that it is always being executed. The method that is executed to acquire

the information needed is *getDeviceContext()*. The device context is stored in the attribute of the user model "session.member.pluginvideo".

```
<TPersonalization>
    ...
    <profile name="ooh:all" condition="">
        <rule type="acquisition" name="Storeplugin" support="12" confidence="100" priority="Medium"
            activation="1/12/02" expires="1/09/03" lastTimeExecuted="2/02/03">
            <periocity periocityValue="ooh:always" method="getDeviceContext()" />
            <action value="session.member.pluginvideo=getDeviceContext()"/>
        </rule>
    </profile>
    ...
</TPersonalization>
```

Personalization Rule(2p):

MODELLING AND GATHERING PROCESS

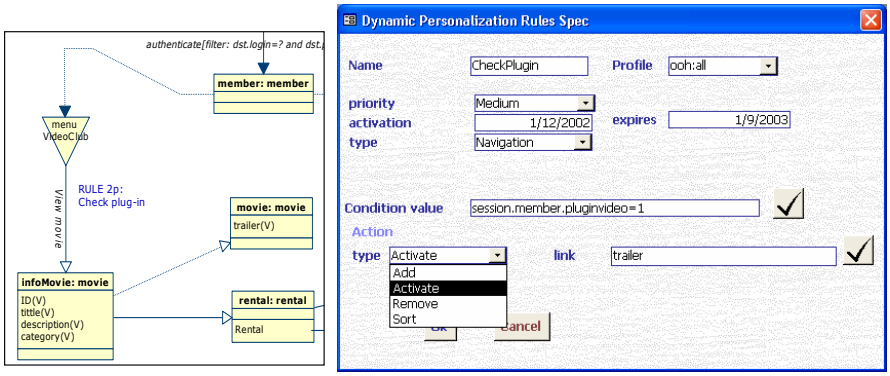

Figure8. *Personalization: rule 2p*

In this image we can see the part of the NAD diagram in which the rule that models this requirement is introduced. When the link *View Movie* is activated the rule is triggered. When this link is double clicked, the box dialog shown in Fig. 8 appears to introduce the needed parameters to model the personalization rule. These parameters are the same that in the rule 1p with the difference that in the action type the values that appear are the corresponding to the navigation rules *Add, Activate, Remove, Sort*. It indicates that if the plug-in is detected, a link to view the trailer will be shown.

XML SPECIFICATION

In this case, the personalization rule to implement this case is a navigation rule because the visualization of a link depends on the device context. We have as a parameter the attribute *plug-invideo* that is a boolean which indicates if the member has the needed plug-in. If the condition is accomplished, in the action is specified the activation of a new link to view the movie trailer.

```
<TPersonalization>
    ...
    <profile name="ooh:all" condition="">
        <rule type="personalization:navigation" name="CheckPlugin" support="12" confidence="100" priority="Medium"
            activation="1/12/02" expires="1/09/03" lastTimeExecuted="2/02/03">
            <params>
                <param name="pluginvideo" value="session.member.pluginvideo"/>
            </params>
            <condition value="pluginvideo=1"/>
            <action type="Activate" value="session.movie.trailer='visible'"/>
        </rule>
```

```
    </profile>
    ...
</TPersonalization>
```

FINAL REPRESENTATION

Figure 9 shows the final appearance of this requirement in case the user has the properly video plug-in. The link 'view trailer' is showed together the information about the movie.

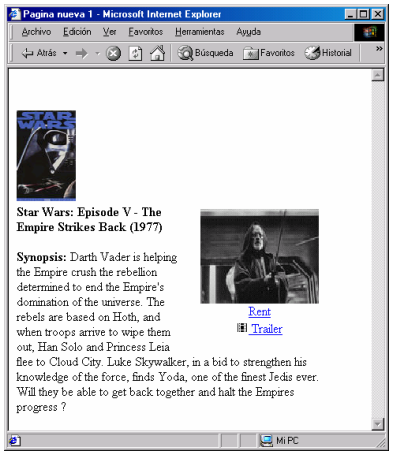

Figure9. *Final Representation: rule 2p*

5.3 Personalization of Presentation

Finally, as an example for personalization of presentation we have the next requirement:
In the case of a user with a deficient vision, the font will be shown in a bigger size.

There is not acquisition rule at this level because this information is captured in an explicit way by the user.

Personalization Rule (3p):

MODELLING AND GATHERING PROCESS

In this image we can see the part of the CLD diagram in which the rule that models this requirement is introduced. When we select from the menu the option 'Plug-ins' $->$'Personalization Properties' $->$ 'Set a Presentation Rule', the box dialog shown in Fig. 10 appears to introduce the needed parameters to model the personalization rule. The defined rule is applied to the selected abstract page. In this case we apply the rule to all the existing pages. The parameters to define the rule are the following:

- *Name of the rule*
- *Profile:* it can be selected from all the existing profiles that have been created, or the special value "ooh:all" that indicates that the rule will be applied to all profiles.
- *Page/s:* it indicates which page/s will be affected by the application of the defined rule. In this case the special value "allPages" is selected, to apply the rule to all the pages in our web site.
- *Priority of the rule:* it can take the following values depending on the priority of execution of the rule : *low, medium, high.*

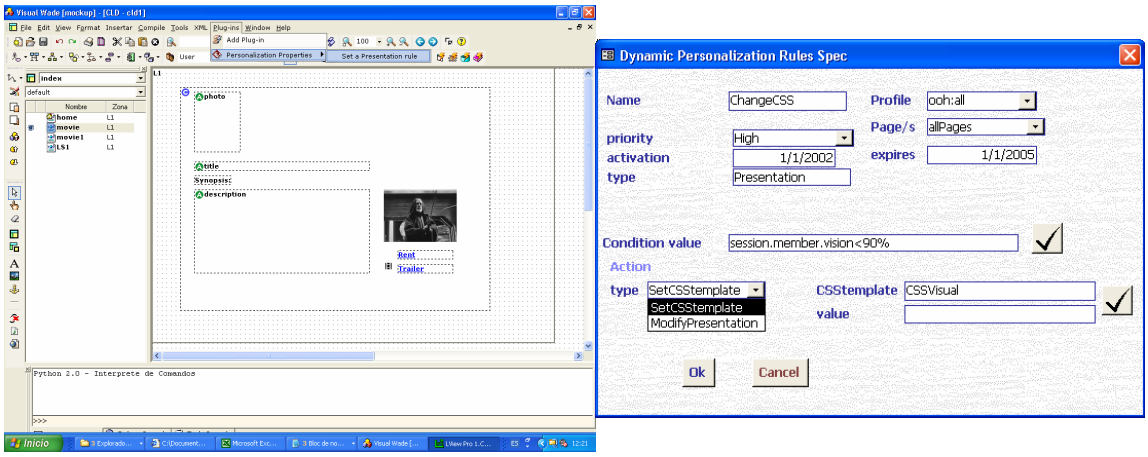

Figure10. *Personalization: rule 3p*

- *Date of activation of the rule*
- *Date in which the rule expires*
- *Type of the rule:* we can model a content or a navigation rule, in this case, the type of the rule is *presentation*.
- *Condition value:* This parameter is an OCL expression which has to be accomplished to trigger the rule. It can be validated by means of an OCL compiler.
- *Action type:* This tag can take as values the actions that a presentation rule can take: *SetCSStemplate, ModifyPResentation*. In this case the value taken is *SetCSStemplate*.
- *Action value:* The action that will be performed when the premises are accomplished. This value can be a CSStemplate or an OCL expression that can be validated. It depends on the action type of the rule.

XML SPECIFICATION

This personalization requirement is represented as a *Presentation Rule* (this type of rule was presented in section 4). In this example, we use a predefined CSS template for a sight-disabled person. In the personalization rule we have a Start event. When the user enters the system the condition will be evaluated and, if the condition is accomplished, the action will be executed.

```
<TPersonalization>
    ...
    <profile id="ooh:all" condition="">
        </rule>
            <rule type="personalization:presentation" name="ChangeCSS" support="12" confidence="100" priority="High"
             activation="1/01/02" expires="1/01/05" lastTimeExecuted="1/01/03">
            <params>
                <param name="vision" value="session.socio.vision"/>
            </params>
            <event type="Start"/>
            <condition value="vision<90%"/>
            <action type="SetCSStemplate" CSStemplate="CSSvisual"/>
        </rule>

    </profile>
    ...
</TPersonalization>
```

FINAL REPRESENTATION

The final result in the storyboard is that the text font is showed in a bigger size. Figure 11 shows the final appearance of the storyboard for this requirement.

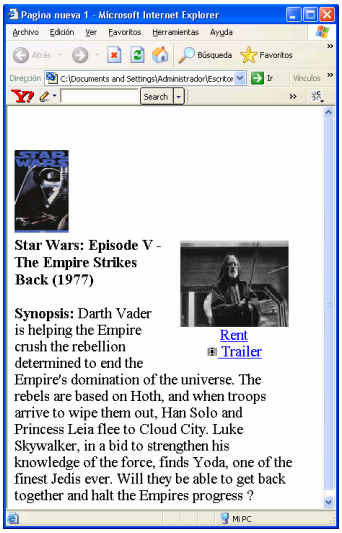

Figure11. *Final Representation: rule 3p*

6 Conclusions and further work

Web Engineering methods have to provide a well-defined software development process by which the community of software engineers can properly design powerful web-based applications in a systematic way. Our purpose has been to address these problems in the context of the OO-H conceptual modelling approach that has been successfully proven for the design of web applications. We focus on how to properly capture the particulars associated to the design of dynamic personalization. In order to achieve this goal, OO-H adds acquisition and personalization rules, which define the suitable semantics for capturing and representing the specific functionality of dynamic personalization. In this way, navigation and presentation models can be compiled to obtain an XML specification that represents the desired dynamic personalization. The final web application is viewed as a composition of a stable and variable part, where the variable part is interpreted by the rule engine to give personalization support. The main benefit is that personalization specification can be modified without need for recompiling the rest of the application modules.

Others relevant contributions of this paper are the following:

1. a user model that describes how the knowledge that the system has about the user is captured.
2. a reference model that provides a way to extend the OO-H metamodel by means of a specific a set of information structures.

OO-H is still defining and cataloguing a set of both navigation and presentation personalization patterns general enough as to guarantee the application reusability. Also, this way to support dynamic personalization is being implemented in the OO-H CASE environment to obtain empirical results regarding the applicability of this work.

References

[1] C. Cachero, J. Gómez, and O. Pastor. Object-Oriented Conceptual Modeling of Web Application Interfaces: the OO-HMethod Presentation Abstract Model. In 1^{st} *International Conference on Electronic Commerce and Web Technologies (ECWEB'00), Greenwhich, London.*, volume 1875, pages 206–215. Springer-Verlag. Lecture Notes in Computer Science, 09 2000.

[2] S. Ceri, P. Fraternali, and A. Bongio. Web Modeling Language (WebML): a modeling language for designing Web sites WWW9 Conference. In *First ICSE Workshop on Web Engineering, International Conference on Software Engineering*, 05 2000.

[3] Harel D. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, 8(3), 1987.

[4] U. Dayal. Active Database Management Systems. In *Proc. 3rd Int. Conference on Data and Knowledge Bases*, pages 150–169, 1988.

[5] I. Garrigós, C. Cachero, and J. Gómez. Modelado Conceptual de aplicaciones adaptivas y proactivas en OO-H. In *II Taller sobre Ingeniería del Software Orientado al Web (JISBD 2002)*, 11 2002.

[6] A. Ginige and S. Murugesan. Web Engineering: an Introduction. *IEEE Multimedia Special Issue on Web Engineering*, pages 14–18, 04 2001.

[7] J. Gómez, C. Cachero, and O. Pastor. Extending a Conceptual Modelling Approach to Web Application Design. In 12^{th} *International Conference on Advanced Information Systems (CAiSE'00)*, volume 1789, pages 79–93. Springer-Verlag. Lecture Notes in Computer Science, 06 2000.

[8] J. Gómez, C. Cachero, and O. Pastor. Conceptual Modelling of Device-Independent Web Applications. *IEEE Multimedia Special Issue on Web Engineering*, pages 26–39, 04 2001.

[9] N. Koch, A. Kraus, and R. Hennicker. The Authoring Process of the UML-based Web Engineering Approach. In *Proceedings of the 1st International Workshop on Web-Oriented Software Technology*, 05 2001.

[10] Carneiro L., Cowan D., and Lucena C. Introducing ADV Charts: A Graphical Specification of Abstract Data Views. In *Proceedings of CASCON'93*, 1993.

[11] W. Retschitzegger and W. Schwinger. Towards Modeling of DataWeb Applications - A Requirement's Perspective. In *Proceedings of the American Conference on Information Systems AMCIS 2000*, volume 1, pages 149–155, 08 2000.

[12] Daniel Schwabe and Gustavo Rossi. A Conference Review System with OOHDM. In *First International Workshop on Web-Oriented Software Technology*, 05 2001.

[13] Olga De Troyer and Sven Casteleyn. The Conference Review System with WSDM. In *First International Workshop on Web-Oriented Software Technology*, 05 2001.

[14] eXtensible Markup Language. http://www.w3.org/XML/, 2000.