

# Modelling Dynamic Personalization in Web Applications

Irene Garrigós<sup>1</sup>, Jaime Gómez<sup>1</sup>, and Cristina Cachero<sup>1</sup>

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 but often these proposals are 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. To model the variable part of the interface logic OO-H has a personalization architecture that leans on a rule engine. Acquisition rules are defined by means of a *User Model* and a *Reference Model* and allows to capture the relevant properties to give dynamic support. Personalization rules are applied to the navigation and presentation level, and are reflected in their corresponding conceptual views. In this way, the interface logic of a web application is viewed as a composition of an 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 [5], help designers to make easier the understanding, development, evolution and maintenance of web applications. These methods are based on new constructors and hypermedial views [7, 10, 1, 11] 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 effort. To achieve this goal, 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)* [6] is extended to support dynamic personalization. We describe how conventional navigation and presentation diagrams are influenced by personalization properties. These properties are captured in form of external files written in XML, that represents the association 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 of this architecture is done 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.

In this way, the interface logic of a web application is viewed as a composition of an 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. In concrete, 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 in function 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 of personalization in function of roles, like OOHDM [10] or WSDM [11].

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 [8] while dynamic aspects of the user interface are modelled with a technique bases on Statecharts [2]. The modelling and design in a conceptual frame allow to have a better understanding of the used mechanisms and discover common features that allow to reuse patterns, components, algorithms or even subsystems [10]. In the WSDM approach [11] different perspectives are defined for the user classes; these are different ways classes 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 [1] or UWE [7].

In the W3I3 approach [1], some aspects of the personalization policies are detected after running the application, the method include mechanisms to externalize the policies and to support their change once the application has been deployed. However, these mechanisms are implemented as a database triggers and are focused for the design of data-intensive web applications.

UWE [7] is centered in the specification of adaptive applications. It insists on personalization features, like the definition of a user model or a stage of definition of adaptive features of navigation in function of preferences, knowledge or tasks that the user has to execute. From our point of view this is the more complete proposal. It provide a formal theoretical framework for dynamic personalization. However, the main problem is the inflexibility of UWE conceptual models. We mean that any change the designer wants to introduce must be supported in UWE framework.

Finally, there are 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 cause reuse problems and difficult maintenance and scalability of the resultant personalized applications.

Once we have presented the state of the art in dynamic personalization and identified the main limitations in the current approaches, we are going to describe how OO-H provides personalization support.

### 3 Personalization Support in OO-H

#### 3.1 Personalization Architecture

The architecture of OO-H has been extended to support dynamic personalization. Figure 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: an 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.

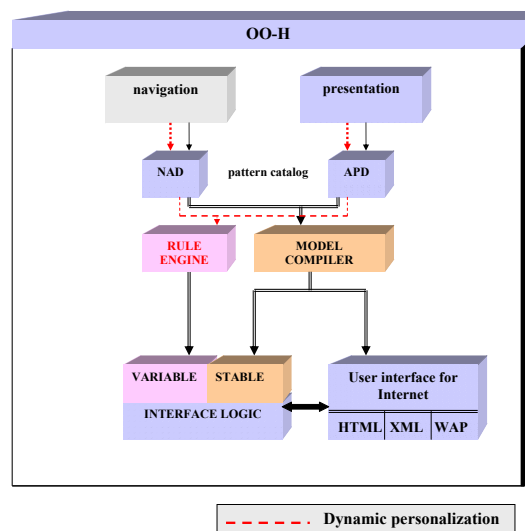


Figure1. OO-H Architecture with dynamic personalization

Modelling the variable and stable parts of the application in a separate way since the first states of the software life cycle, the dynamic nature of the ubiquity of web applications can be better treated, as the reusability and execution of changes. The justification of this statement is similar to the reasons argued to separate business policies from object oriented applications. [9]. The support of this architecture is gotten by a *Reference Model* that allows to capture the relevant properties of personalization and is presented in the section 3.3. Moreover, a *User Model* must be defined to support the personalization requirements. It is 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 [7].

In OO-H the user model is a Class Diagram that complements the Conceptual model. It captures information about the features that the system, which we are modelling, believes the user has. This features can be preferences or interests, general knowledge, experience...etc. A user model can never

be completely accurate, it is usually a rough approximation. Though, an incomplete user model can be useful. The improvement of the user model has to be compared with the effort that has to be made for it [7]. 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 in the concepts of user and role, the same that in another hypermedial approaches [1]. To allow personalized views to the user OO-H provides a basic user model, which has a User Class, from which inherits in an implicit way all the domain classes stereotyped as <<actor>>, that define the possible roles or profiles of a user in the application. A user can not have a role associated, in this case he/she would be 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 to the 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 in function of 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 he/she wants to fit with ability of personalization. Starting from there, OO-H allows the extension of this repository with the particular features that requires the application, therefore, it isn't a closed frame. First version of this repository was presented in [4]. 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 purposes of this paper we must pay more attention to the part B of the figure 2, it is, to the association rules. A new type of rule has been added to this repository: the presentation rule (see Fig. 2).

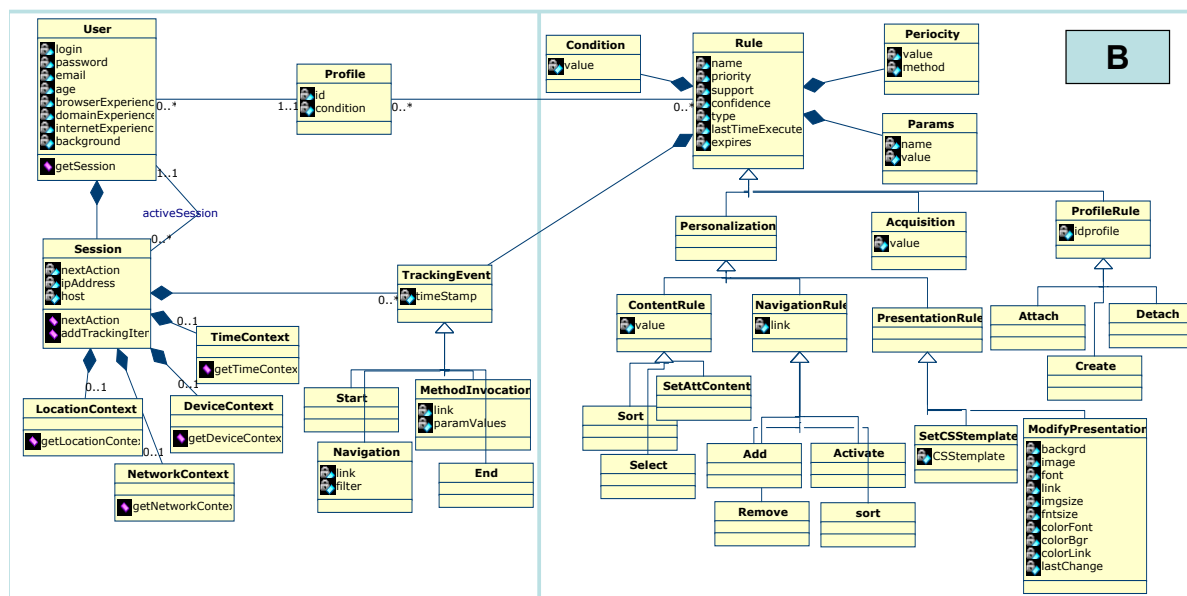


Figure2. Personalization Framework

The **Association Rules** are presented in profundity in the following section. Like 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 this 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 two 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, in function of 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 whether we want to use a predefined template of presentation or we want to modify an specific value. We have a set of CSS templates, designed to specific cases (as for disabled persons) 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 an specific conceptual model (see figure 3). In the rules, it can be chosen a predefined template or modify specific features of the Presentation Class. We can see an example in the 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 a XML schema [12]. Is important to note that the execution architecture of the generated applications from OO-H models allows to modify and reprocess this schema without recompile the rest of modules. In the schema the different XML elements correspond to classes/attributes of the frame presented in the Fig. 2.

All the rules are ECA (Event-Condition-Action) rules [3]. 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 can take a rule vary depending on the type of the implemented rule, like we have seen.

In the next section there is shown an example of the three types of dynamic personalization that can be modelled in OO-H: Personalization of Content, Navigation and Presentation. We'll use as information system a video club in Internet with 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 properly 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, like we said in the 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 [7]. In this case it may be as the model seen in the figure 3.

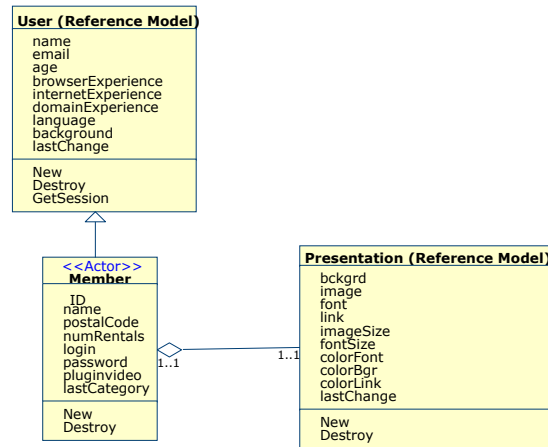


Figure3. User Model

This user model results of the connection of the *User* class, from the reference model, with the existing class *Member*, from the conceptual model, and treat 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:

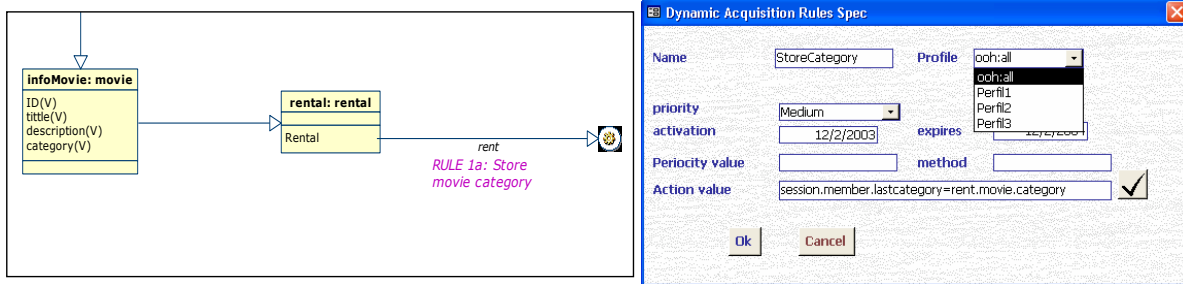
- *Modelling and Gathering process*: In this phase is shown how the requirement is introduced in the NAD/CLD diagram.
- *XML specification*: Here is shown a XML specification that supports the requirement.
- *Final Representation*: Here is shown the storyboard of the corresponding requirement.

## 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):**



**Figure4.** Acquisition: rule 1a

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 the figure 4 appears to introduce the needed parameters to model the acquisition rule. These parameters are:

- 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
- Periodicity value: the periodicity in which the rule is executed if is the case of a proactive rule.
- Method that is used to acquire the information (if 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 a 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 to 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 the figure 5 appears to introduce the needed parameters to model the personalization rule.

These parameters are:

- Name of the rule

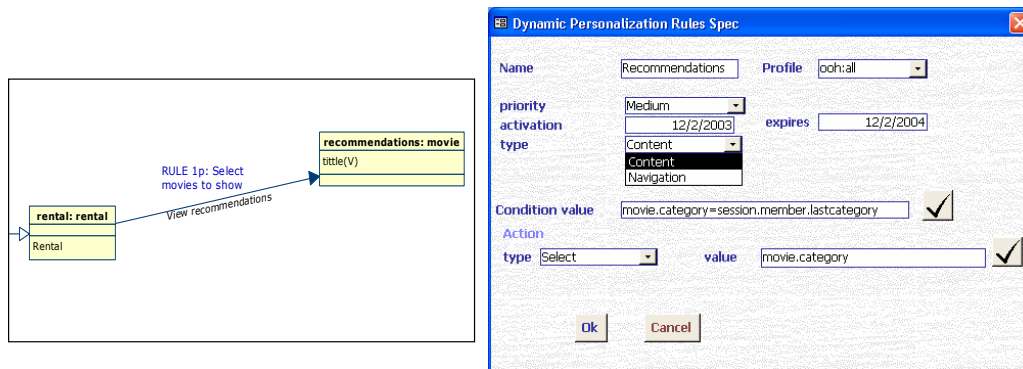


Figure 5. Personalization: rule 1p

- *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 interpreter.
- *Action type*: This tag can take as values the actions that a content rule can take: *select, sort, setattcontent*. In this case the value taken 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 SPEC

In this case, we have content personalization, because we have to show or hide an 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.

## 5.2 Personalization of Navigation

The requirement of the Personalization of Navigation is the next one:

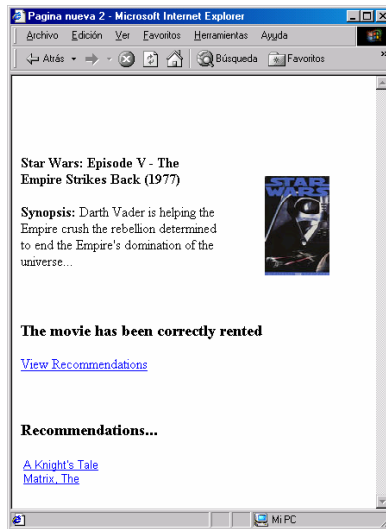


Figure6. Final Representation: rule 1p

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

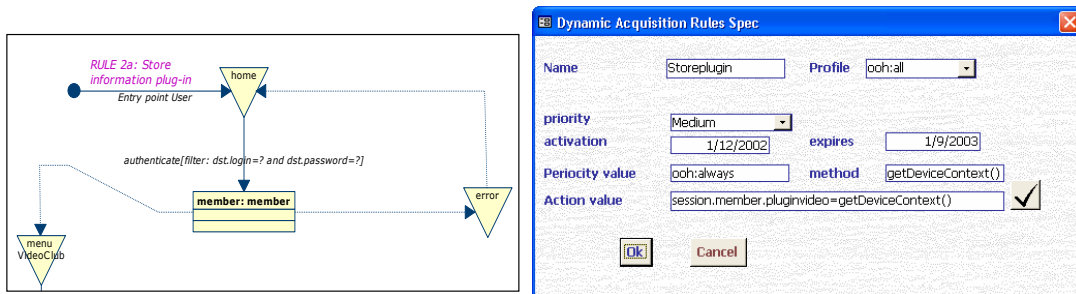


Figure7. Acquisition: rule 2a

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

### XML SPECIFICATION

In this case the XML spec 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 and 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()" />
  </rule>
</profile>
</TPersonalization>
```

```

    <action value="session.member.pluginvideo=getDeviceContext()"/>
  </rule>
</profile>
...
</TPersonalization>

```

### Personalization Rule(2p):

## MODELLING AND GATHERING PROCESS

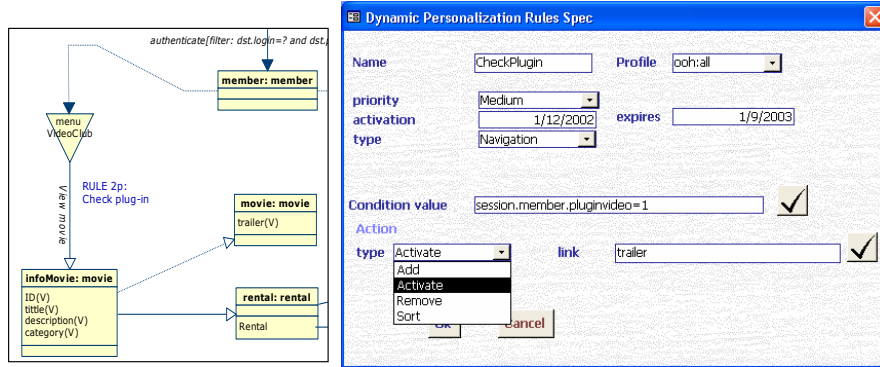


Figure8. Personalization: rule 2p

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 the figure 8 appears to introduce the needed parameters to model the personalization rule. These parameters are the same that in the rule 1p and indicates that if the plug-in is detected, a link to view 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 *pluginvideo* 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.

### 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.*

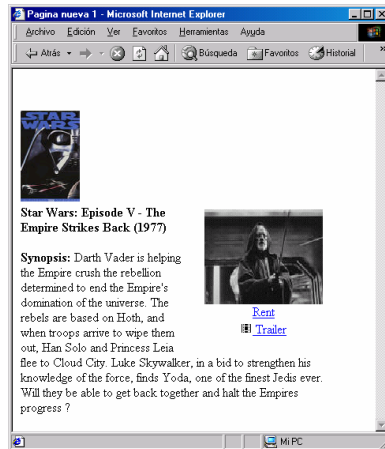


Figure9. Final Representation: rule 2p

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 the figure 10 appears to introduce the needed parameters to model the personalization rule. The defined rule is applied to the abstract page that is selected in the moment of setting the rule. When the rule is applied to the home page, by default is applied to all the pages. In this case we apply the rule to the home page.

The parameters to define the rule are the following:

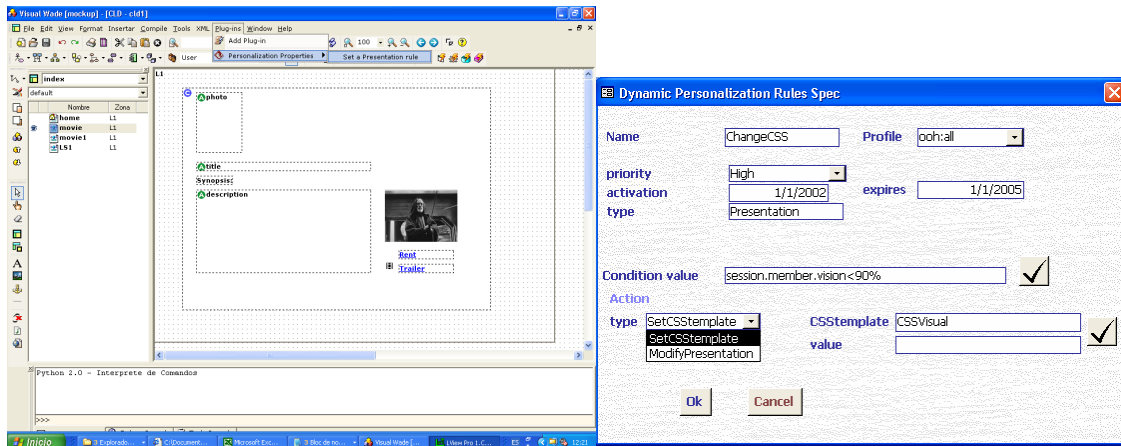


Figure10. Personalization: rule 3p

- 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*
- *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 interpreter.
- *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 CSS template or an OCL expression that can be validated. It depends on action type of the rule.

### XML SPECIFICATION

This personalization requirement is represented as a *Presentation Rule* (this type of rule was presented in the section 4). In this example, we use a predefined CSS template for a disabled sight person. In the personalization rule we have a Start event. When the user enters to 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 story board is that the text font is showed in a bigger size. Figure 11 shows the final appearance of the storyboard for this requirement.

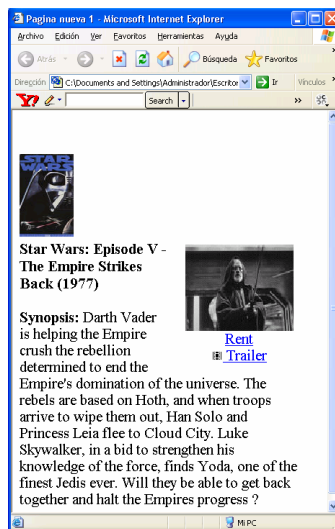


Figure11. Final Representation: rule 3p

## 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 proven successful 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 semantics suitable 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 an stable and variable part, where the variable part is interpreted by the rule engine to give personalization support. The main benefit is that personalization spec can be modified without recompile 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 in the applicability of this work.

## References

- [1] 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.
- [2] Harel D. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, 8(3), 1987.
- [3] U. Dayal. Active Database Management Systems. In *Proc. 3rd Int. Conference on Data and Knowledge Bases*, pages 150–169, 1988.
- [4] 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.
- [5] A. Ginige and S. Murugesan. Web Engineering: an Introduction. *IEEE Multimedia Special Issue on Web Engineering*, pages 14–18, 04 2001.
- [6] 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.
- [7] 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.
- [8] Carneiro L., Cowan D., and Lucena C. Introducing ADV Charts: A Graphical Specification of Abstract Data Views. In *Proceedings of CASCON'93*, 1993.
- [9] 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.
- [10] Daniel Schwabe and Gustavo Rossi. A Conference Review System with OOHDm. In *First International Workshop on Web-Oriented Software Technology*, 05 2001.
- [11] Olga De Troyer and Sven Casteleyn. The Conference Review System with WSDM. In *First International Workshop on Web-Oriented Software Technology*, 05 2001.
- [12] eXtensible Markup Language. <http://www.w3.org/XML/>, 2000.