A Reusable Personalization Model in Web Application Design

Irene Garrigós, Jaime Gómez
Universidad de Alicante
Campus de San Vicente del Raspeig, Apartado 99 03080
Alicante, Spain
{igarrigos,jgomez}@dlsi.ua.es

Peter Barna, Geert-Jan Houben
Techniche Universiteit Eindhoven
PO Box 513, NL-5600 MB
Eindhoven, The Netherlands
*{*P.Barna, G.J.Houben *}@tue.nl*

Abstract

Personalization of web sites has become an important issue in web modeling methods. Many of these methods use similar approaches to specify personalization. However, even with similar (personalization) requirements the resulting implementations differ. The consequence is that we cannot reuse the same (personalization) specification. This paper presents a generic extension for existing conceptual modeling approaches to address the particulars associated with the design and specification of dynamic personalization. We allow the designer to specify, at design time, how the website will be personalized (at runtime). We claim that the personalization specification can be mapped to different web modeling methods so we can define this specification once and reuse it for different (development) environments. For this purpose, a high level (generic) language based on rules (Personalization Rules Modeling Language) is presented. Finally, we describe how PRML rules can be easily mapped to the rule representation for a commercial rule engine.

1. Introduction

Web idiosyncrasy introduces new challenges for the design process that go further than the specification of the navigation map, and that include aspects like the need for continuous evolution, and the treatment of a heterogeneous audience (that implies different user's needs, goals, preferences, capabilities etc. in our website). In this context the personalization of websites has become an important issue in web modeling methods [6] [8] [9] [11] [12] [15], due also to the effect of the diversity of personalization policies over all the development cycle of applications. Many of these methods tackle very similar problems and use conceptually similar techniques (personalization model based on rules, user model, etc). Despite of this, even with the same (personalization) requirements the final implementations differ. This is why we cannot reuse the same (personalization) specification. When adaptation rules (from personalization model) are specifically defined for an application environment the re-usability of rules is a big challenge [14]. Only when rules management is separated from application development personalization will achieve maximum flexibility.

It is obvious that many existing approaches offer powerful specification means and tools for personalization. However, in most cases they are not portable, typically use different specification techniques and are aimed and implemented for different design methods. In this paper we offer a solution that preserves the advantages of the existing approaches, and in our opinion it also shows a good balance of expressive power (aimed to personalization) and portability – its ability to be (automatically) mapped to rather different conceptual modeling methods using different specification techniques. This solution is based on a high level rule language called PRML (Personalization Rules Modeling Language) that the designer can use to specify (at design time) the personalization to be performed at runtime. The specification of personalization is separated from any other application functional specification, so it can be specified once and used for different (development) environments. The personalization specification can be mapped (implemented) to different existing web design approaches. To demonstrate our claims, we discuss its mapping to OO-H [7] [8] and Hera [9] [18] representing rather different approaches to web design.

The paper begins describing backgound in section 2. Section 3 explains how, in a web design model context, personalization is applied. An example specified in the methods studied in the paper demonstrates two different modeling techniques where personalization is modeled. In section 4 the PRML language is presented and the mapping from PRML to OO-H and Hera is described. In the next section we show how PRML rules can be easily mapped to the rule representation for a commercial rule engine. Finally we present conclusions and further work.

2. Background

Adaptation rules are widely used in the literature for specifying adaptive websites. There are different kinds of adaptation rules depending on the object of the adaptation: we focus on the user-related adaptation rules (personalization rules). A personalization rule will state when concrete users see certain pieces of content and how specific functionality is presented. Some modeling approaches have their own language for defining rules.

For example, for describing navigation goals WebML [6] uses the WBM formalism based on a timed state-transition automata. This formalism combined with WebML form a high-level Event-Condition-Action paradigm. In WebML an event consists of a page request, the condition is expressed as a WBM script and represents a set of requeriments on the user navigations, and the action is expressed as a WebML operation chain and can specify adaptivity actions. In WSDM [3][4], in the context of audience driven design, a language is provided to help the designer to define adaptive behavior, focusing on the adaptation for a group of users by adapting the structure of and the navigation possibilities in the website. WSDM doesn't support personalization, but this language could be used to personalize. UWE [11] in the context of an OO method, define adaptation rules as OCL expressions. A rule is modeled as a class Rule that consists of one condition, one action and attributes. Rules are classified according to their objectives into: construction rules, acquisition rules and adaptation rules.

In WUML[10], in the context of ubiquitous computing, customization rules allow the designer to specify the adaptation to perform when a certain context is detected. These rules are specified within an UML annotation, using the stereotype <<CustomizationRule>>. The specification of such a rule comprises a unique name, a reference to one or more requirements, and an ECA triplet. The annotation is attached to those models elements being subject to customization. Thus the customization rules can be attached to any web application modeling using UML as the basic formalism.

However, when personalization rules are specifically defined for an application environment we find some problems: the re-usability (in different approaches) of the personalization strategy is (almost) impossible. Moreover, due to the heterogeneous ways to personalize in the different methods, it is difficult to compare them by the (level, kind of) personalization they support.

Also, 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. Some of them also allow to personalize in basis of rules The main problem of these approaches is the low abstraction level that causes reuse problems and a difficult maintenance and scalability of the resultant personalized applications.

3. Contextualization of Personalization in Web Design Methods

In the context of web design methods, the existence of conceptual models independent of the final implementation language makes easier the personalization (adaptation) of the web applications to the environment changes, reusing the information captured during the process of analysis and design. Focusing on the conceptual modeling phase, every design method has three main layers: a Domain model, in which the structure of the domain data is defined, a Navigation model which defines the structure and behavior of the navigation view over the domain data. Finally, the Presentation Model defines the layout of generated hypermedia presentation. To be able to model adaptation/personalization at design time two new models are added (to the set of models): (1) A Personalization Model, in which personalization rules are defined to store information needed to personalize and to specify different personalization policies. (2) A User Model, which allows to store data needed for personalization besides the beliefs and knowledge the system has about the user. This information builds the user profile and it is needed to base the personalization on.

As already mentioned, we propose to create a reusable personalization model. In this paper we are going to focus on two modeling methods different enough to show the generality of our proposal. The OO-H Method [7] [8] is based on the OO paradigm that provides the designer with the semantics and notation necessary for the development of web-based interfaces. Hera [9] [18] is a model-driven methodology for designing web applications. Both methods have three layers for designing a web site. In this paper we (only) focus on the personalization of the structure of the website (not in presentation details). In the next subsections we define the models of the conceptual modeling phase, and compare these two methodologies.

We study a (simplified) book store shopping basket system as a running example. Consider the following personalization requirement: *we want the user to be offered a list of books written by authors in which user has certain interest degree.*

3.1 Domain Model and User Model

In figure 1 we see the DM and the UM for OO-H for the running example. The UM is represented as part of the DM. These models are defined (in OO-H) as UML compliant class diagrams. Concepts are represented by classes with attributes. There are associations between them representing Concepts relationships. We can store or update the information of this model by means of ECA [5] rules as we will show in section 4.

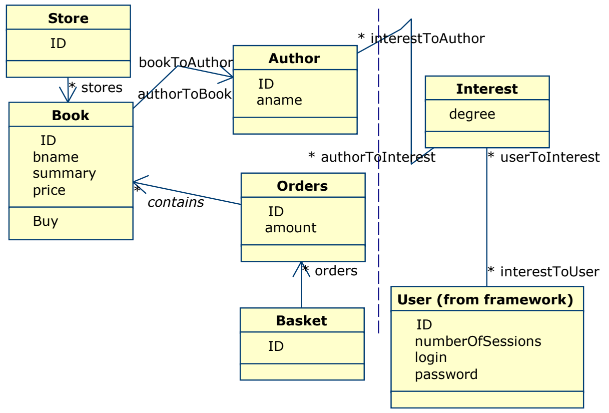

Figure 1 DM-UM for OO-H (expressed in UML)

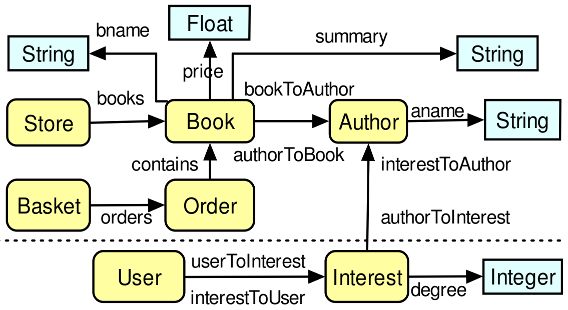

Figure 2 DM-UM for Hera (expressed in RDFS)

In figure 2 we can see the Hera DM and UM for the same example. In Hera, all models are represented with RDFS[2]. Ovals represent Concepts and rectangles represent literal Attributes. In both methods, for the purpose of the personalization requirement defined, we store in the updatable UM the user's interests on authors.

3.2 Navigation Model

A generic Navigation Model (NM) will be composed of Nodes and Links. Nodes are (restricted) views of domain Concepts. Each Node has associated a (owner) Root Concept from the DM. Browsing Events are associated to the Links of the NM. A NM describes a navigation view on data specified by DM (and possibly also UM). Moreover it also captures the functionality (personalization) of the designed application.

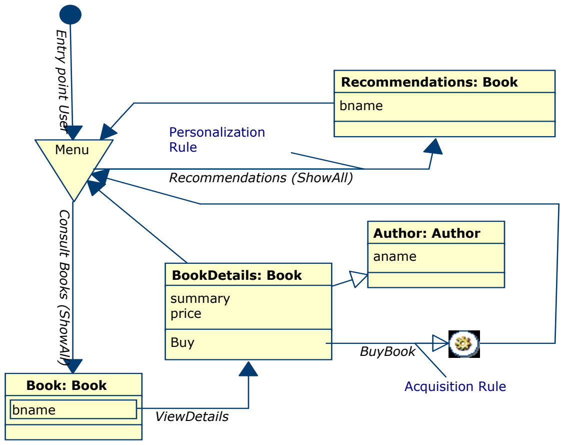

Figure 3 OO-H Navigation Model

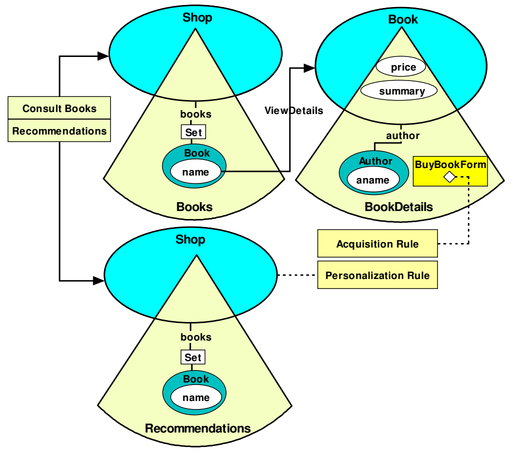

Figure 4 Hera Navigation Model

In figure 3 we can see the NM for OO-H, and in figure 4 for Hera for the running example. In both cases, the final website will have as a starting point: the home page where we can find a collection of links: "Consult Books" and "Recommendations" in this example. If the user navigates through the first link (Consult Books), he will find a list with all the books in a new page. This set of books is an indexed list, so the user can click in one of the books names to view the details of each of the books. When the user clicks on "Recommendations" personalized book suggestions should appear (based on the information from UM). To fulfill this personalization requirement (Users will see recommendations of books based on their interests in authors) we need to define a personalization model to adapt this NM. It will be explained in section 4.

The OO-H NM is composed by navigational classes, which represent views of the classes of the domain model (the above defined Nodes). In each Node we have defined the Root Concept from DM (or UM) attached to it by the notation: "RootConcept:Node". In the Hera NM the ovals with the slice shapes represent navigation Nodes. Every slice is based on a DM (or UM) Concept (Root Concept) indicated by the title on appropriate oval. Overlaying areas of slices and ovals contain attributes of the (root) concept (e.g. price in Books.Detail) and attributes of related concepts are located in the slice areas not overlapping with ovals (e.g. aname in Book.Details). Arrows represent links that have attributes as anchors and slices as targets. For a more extensive description of OO-H and Hera NM we refer the reader to [8] and [9] respectively.

In the next section we introduce the Personalization Rules Modeling Language to specify a personalization strategy and explain how it can be mapped to OO-H and Hera methods.

4. Personalization Rules Modeling Language

The rules of the personalization model will be defined using an effective, simple and easy to learn language. As already mentioned, the purpose of this language is to help the web designers to define all the rules required to implement a personalization strategy. The basic structure of a rule defined with this language is the following:

When *event* **do**
    If *condition* **then** *action* **endIf**
endWhen

The rules we can define with this language are ECA [5] rules. A rule is formed by an event, the body of the rule, a condition (optional) and an action to be performed. For satisfying a personalization requirement we have to define how to obtain the required knowledge to personalize (acquisition rule). This information will be stored in the UM. Also, we have to define the effects this personalization causes in the system (personalization rules). These rules use the information from the UM to describe adaptation actions. Depending on the object of the adaptation two kinds of personalization rules are considered: navigation rules (to alter navigation), and content rules (to add/remove/adapt content). Due to space limitations we can only show an excerpt of the BNF specification of PRML. Symbols and words in bold denote keywords. (In the paper for clarity reasons rules have been simplified omitting the features of the rule, e.g. priority, expiration date, etc.).

```
<rule>::= <features> When <event>do <body>
endWhen
<body>::= [<periocity>] [If ( <condition>) then]
<action> [endIf]
<event>::= (Start | <navigation> |
<methodInvocation> | End )
<navigation>::= Navigation. <link>
<methodInvocation>::= MethodInvocation. <link>
<periocity>::= SetPeriocity <operator> (always |
everyday | eachmonth | <userdefined>) [1]

<condition>::=(<relexpression> [<booloperator>]
{<relexpression>} )
<relexpression>::= <operand> <operator>
(<value>|<operand>) | not <relexpression>
<operator>::= < | > | = | <= | >= | <>

<action>::= <SetContent> | <Show> | <Hide> |
<Sort> | <SortLinks>
<SetContent>::= SetContent(<attribute>,
<content>)
<attribute>::=<attname>
<attname>::=<class> . <attrib>
<content>::=<attname> | <value>
<Show>::= Show(<attname>)
<Show>::= Show(<link>)
<Hide>::= Hide(<link>)
<Sort>::= Sort <class>  orderBy ASC | DESC |
<attname>
<SortLink>::= SortLinks <link> orderBy ASC |
DESC | <link>
```

For the requirement considered in the running example we need the following rules: We need an acquisition rule to update the user's interests (in the UM) when the user consults a specific book. We also need a personalization rule that will do the following: "When the user clicks on the link 'Recommendations' we show the books of authors with an interest degree > 100" (checking the updatable value from UM).

The three parts that form a rule (event, condition, action) are explained in next subsections, by means of the running example, to illustrate the use of the PRML

4.1 PRML: Event Specification

First, we consider the events. We want an active system that monitors situations of interest to personalize and to trigger the proper response when one of these

[1] Periocity states a certain interval of time in which the rule condition has to be checked.

situations occurs. This response is specified by an action in a rule, so to trigger that action we need events. Each time the user activates a link (activelink), the associated event is launched causing the evaluation of a rule. When a link is clicked a node is also activated (activenode). We consider the following types of browsing events:

- *Navigation event:* implies the activation of a navigational link and the activation of a node (i.e. the new page resulting of the navigation). Links have a parameter indicating the instance or set of instances that are going to be shown (and the navigation pattern, i.e. *showall, index..*) in the activated node. When the link is activated the rule/s attached to this event is/are triggered passing to the rule/s this parameter to build, in basis of this information, the active node. We need the data before the node resulting from navigation is generated (in order to use those data in the rules and build the adaptive web page). This information is passed in the navigation event (when possible and needed) as a parameter[2]. This parameter can be a simple parameter (when the data on the webpage is the instance of a concept of the domain model) or a complex parameter (when the data of the webpage is a set of instances of a concept of the domain model). In PRML we can write: **When** *Navigation.activelink(NM.activenode parametername)* **do** to express this event passing a single parameter, and **When** *Navigation.activelink(NM.activenode* parametername)* **do** when we pass a complex parameter. In the running example we could express: **When** *Navigation.ViewDetails(NM.Book book)* **do** to specify the event of a rule triggered when the user browses the link "ViewDetails". As a parameter we add the active node "Book". In the following case we pass a complex parameter when the user clicks on the "ConsultBooks" link: **When** *Navigation.ConsultBooks(NM.Book* bookset)* **do.**
- *Method Invocation event* implies the invocation of a method defined in the website. To express this event in PRML we can write **When** MethodInvocation.*activelink(NM.activenode)* **do.** In the *Method Invocation* event we also pass a parameter containing the instance of the root concept of the node containing the invocation to the method, this parameter can be also simple (i.e. an instance) or either complex (i.e. set of instances). If we would like to add a rule when the user invokes the service *Buy Book* in PRML we would express it as follows (simple parameter): **When** MethodInvocation.*BuyBook(NM.Book)* **do.**
- *Start event* is associated with the entrance of the user in the website (i.e. the start of a session). In PRML this event is expressed as follows: **When** *SessionStart* **do**.
- *End event* implies the exit of the user from the website (i.e. end of a session). In PRML we can express this event as follows: **When** *SessionEnd* **do .**

Mapping *Events* to OO-H and Hera

In OO-H the events are associated with the different types of link in the NM. *Navigation events* are associated with navigational links in the navigation diagram (see fig. 2 , e.g. View Recommendations, Consult Books). *Method invocation events* are launched when the user clicks on links that invoke a service (*service* links*,* e.g. *Buy Book*). The *Start event* is associated with the *Entry point* link in the navigation diagram. The *End event* is triggered after certain inactivity of the user in the system or when the user clicks activates an *exit* link.

In Hera events are associated either with links or with processing of forms (user inputs). A concrete link event corresponds to the instantiation of its target slice, and a form processing event corresponds to the execution of a data (manipulation) query. Another events are associated to the instantiation of a browsing session (for a single user), and its expiration. Hence, we can consider the following event types analogous:

- *Start event* is *Session instantiation* in Hera,
- *End* event is Session expiration in Hera,
- *Navigation* event is Slice instantiation, here the type and data instance of the source slice (the slice where the link anchor was activated) are default session parameters (sliceid, conceptid), and
- *Method invocation* event is Form processing in Hera, since the application logic is in Hera provided by form processing queries.

4.2 PRML: Condition Specification

Once rules are triggered, only if a condition is satisfied an action is performed. In the case of personalization rules the condition is based on information stored in the UM. In PRML to indicate that we are accessing data of that model, we add the prefix "UM" before the path to access that information. Conditions can be based on user-specific information (independent of the domain, e.g. we consider user characteristics, user context, etc) or on domain dependent information (e.g. number of clicks on a certain link). In the example, in the condition of the personalization rule (once the rule has been triggered) we check the interest degree in a certain author (i.e. greater than 100, the condition is based on domain dependent information). For this purpose we need to check all the instances of the UM concept "interest". In PRML it is expressed as follows "**If** (UM.userToInterest.degree>100)

[2] All the parameters have the prefix NM to indicate that they come from the Navigation Model.

then"[3]. We use path expressions to access the data of the condition of a rule of the type: "ConceptRelationship[4].Attribute". (In the case of accessing the information of the UM, the path expression will start with the User concept, if we have to start from any other class the path expression will start with the name of that class).

Mapping *Conditions* to OO-H and Hera

The mapping of the *conditions* to OO-H and Hera methods will be done in conjunction with the mapping of the actions. The reason for this is that PRML rules may contain implicit bindings of expressions in the condition and action parts. For instance in the excerpt of a rule:

If (UM.userToInterest.degree>100) **then**
    Show(UM. userToInterest.interestToAuthor.authorToBook.bname)
endIf

only these instances of the Interest class given by the condition expression are taken into account in the action part (are shown). This is a powerful mechanism, but it also means that we need to consider mapping of whole condition-action parts into different methods rather than to map the conditions and actions separately. We will see it at the end of the next subsection.

4.3 PRML: Action Specification

The action part of a rule specifies the operations to be performed when the rule is triggered and its condition is satisfied. The action, in case of the personalization rules, will specify the adaptation action to perform. In the acquisition rules, we will specify which information is stored in the UM. We show now how to adapt the content and the navigation using this language in the context of the running example.

ª **CONTENT**

For adapting the content we can perform several actions on it. We can, for example, set new content to an attribute. For instance, the acquisition rule needed for the shopping basket example is going to change the content of an attribute from the UM.

```
When navigation.ViewDetails(NM.Book book) do
If(book.bookToAuthor.aname=UM.userToInterest.Ito
Author.aname) then
SetContent(UM.userToInterest.degree,UM.userToInt
erest.degree+10)
  endIf
endWhen
```

This rule is triggered when the user activates the link "ViewDetails". It updates the interest degree on the author of the consulted book using the SetContent statement in which we specify the attribute that we want to modify, and the value or formula that calculates the new value. This rule compares implicitly each of the instances of the class interest with the consulted instance (to properly update the value).

As explained before, we need to keep the data from the navigation previous to trigger the event, to use that information in the rules. For this purpose, in this example in the event of the rule we pass as a parameter the visited instance of the class Book (from the NM). That is why this parameter has the prefix "NM". The rest of the data of this rule have the "UM" prefix, indicating that information is stored in the UM.

•To adapt the content, we could also select the content to show from a class, or depending on a certain condition. The personalization rule of our running example shows this type of content personalization.

```
When navigation.Recommendations(NM.Book* books)
do
    If
(books.booktoAuthor.authortoInterest.degree>100)
and (UM.userToInterest.interestToAuthor.ID=
books.booksToAuthor.ID) then
            Show(books.bname)
    EndIf
endWhen
```

This rule is triggered also by a navigation event, when the user clicks on the link "Recommendations". When the user activates that link, book titles of the authors with interest degree greater than 100 (specified condition) are shown.

We could also sort the content by a certain value, (specifying if the order is ascendant or descendent), as an example we could sort the books by book name:

```
When navigation.ConsultBooks(NM.Book* books) do
    Sort books orderBy ASC books.name
endWhen
```

The rule is triggered when the user goes through the link "ConsultBooks" and sorts the books ordered by book name in an ascendant way. This rule has no condition so when the user activates that link, book titles are sorted.

ª **NAVIGATION**

For adapting the navigation we can perform diverse actions over the links. We can sort the links on a page, and we can show or hide links. For example we could

[3] The loop to check all the instances is not expressed in the rule, it is done implicitly.

[4] The ConceptRelationship statement can be repeated 1 or more times depending on what we need to access.

hide the link “ViewRecommendations” for users that have not consulted any book yet.

```
When navigation.ViewRecommendations(NM.Book*
books) do
    If (NM.ViewDetails.nclicks>0) then
        Show(NM.ViewRecommendations)
    endIf
endWhen
```

Mapping *Conditions* and *Actions* to OO-H and Hera

As already explained in previous section, the mapping of the conditions and actions is done together. The specification of conditions and actions in PRML can be seen as a declarative query language designed for personalization purposes. This approach allows for transformation of PRML rules to different query languages ranging from OO (like OCL) to relational (like SQL), or Semantic Web query languages (like RQL, SeRQL).

To avoid possible problems with automatic translation of PRML rules (condition and action parts) to methods relying on declarative query languages (OO-H and Hera among others) we apply to PRML rules the following restrictions:

- *Action* parts can contain only one operation, because is not always possible to implement an arbitrary sequence of operations by a single query or automatically derive a postcondition of such a sequence, and
- *Action* parts cannot activate other rules (they don’t trigger events); this restriction would facilitate the deployment of PRML and would avoid risks of infinite loops and deadlocks.

In general, PRML actions contain instance selection (filtering), instance sorting, and instance modifications. Let’s see how PRML mapping is done to OO-H and Hera methods. In case of OO-H, conditions are going to be mapped to OCL [17] expressions. PRML actions are translated into methods of OO-H action classes of the Personalization Model (represented as a class diagram). In this diagram rules are implemented as subclasses of the Rule class, including specializations for different kinds of actions like *SetContent* for modification, or *Sort*, *SortLinks* for sorting. All these OO-H PM classes inherit of the main class *Rule*, which has an abstract *action()* method which must implement the PRML actions. The concrete PRML operations specified by the designer are then converted to OO-H (plus OCL) specification as it is described in Table 1.

Table 1 Overview of implementing selected PRML operations in OO-H and OCL

PRML	Implementation in OO-H
Operation	
Show	• PRML condition and action: applies to precondition of the *action()* method of the *Show* class.
Sort	• PRML condition: precondition of the *action()* method of the *Sort* (*SortLink*) class. • PRML action: postcondtion of the *action()* method of the *Sort* (*SortLink*) class.
SetContent	• PRML condition: precondition of the *action()* method of the *SetContent* class. • PRML action: postcondtion of the *action()* method of the *SetContent* class.

An example of a more complete personalization rule for presenting books in authors, in which the user is most interested is next:

```
When navigation.Recommendations(NM.Book* books)
do
If (UM.userToInterest.degree > 100) and
   (UM.userToInterest.ItoAuthor.ID =
books.booksToAuthor.ID) then
Show(UM.userToInterest.interestToAuthor.authorTo
Book.bname)
 endIf
endWhen
```

The mapping to OCL code for this rule (action and condition) is:

```
navigation.Recommendations(b:
Collection(NM.Book))
Context Recommendations_Show::action()
pre: b->select( i : UM.userToInterest | i.degree
> 100 and
i.interestToAuthor.authorToBook.bname-
>includes(b.bname))
```

Each OCL expression is written in the context of a specific instance, concretely of instances of UM/NM classes appearing in expressions in condition and action parts. In an OCL expression, the reserved word self is used to refer to the contextual instance.

In this example, the precondition determines the selection of the books instances satisfying the PRML rule condition (based on the user interest degree). It applies to the *action()* method of the appropriate Rule subclass. Let’s see now an example of how to map to OCL an acquisition rule:

```
When navigation.ViewDetails(NM.Book book) do
If(book.bookToAuthor.aname=UM.userToInterest.int
erestToAuthor.aname) then
SetContent(UM.userToInterest.degree,UM.userToInt
erest.degree+10)
```

```
endIf
endWhen
```

the OCL code would be as follows:

```
navigation.ViewDetails(b: Book)
Context SetContent_VD::action()
pre: b.author.aname =
UM.userToInterest.interestToAuthor.aname
post: UM.userToInterests.degree =
UM.userToInterest.degree@pre + 10
```

The precondition of the appropriate subclass of the Rule class (representing the concrete rule) constrains the *action()* method by applying the original PRML rule condition, and the postcondition determines how the user interest is updated.

In Hera PRML rules are implemented as (SeRQL[13]) queries that can manipulate or select data. Table 2 shows an overview of few selected PRML operations and their implementation in Hera.

Table 2 Implementation of selected PRML operations in Hera

PRML Operation	Implementation in Hera
Show	A selection query associated with an appropriate slice.
Sort	A selection query associated with an appropriate slice containing an ordering directive.
SetContent	A data manipulation query attached to a form submission. Most typically a couple of REMOVE and CONSTRUCT queries (SeRQL does not support UPDATE yet).

In general,,in SeRQL, queries have the form:

SELECT | CONSTRUCT | REMOVE action-variables
FROM Path-expresions-with-variable-bindings
WHERE Transformed-condition

The first clause depends on the action part (e.g. SELECT for filtering, for changing values first REMOVE query followed by CONSTRUCT query, etc.) and the action-variables part contains variables bound with path expressions appearing in the action part of rules. The Path-expresions-with-variable-bindings part contains path expression appearing in action and condition parts and variables. The WHERE clause provides the evaluation of the condition part.
Let's see a mapping to Hera for the following rule:

```
When navigation.Recommendations(NM.Book* books)
do
    If (UM.userToInterest.degree > 100) and
      (UM.userToInterest.ItoAuthor.ID =
books.booksToAuthor.ID) then
Show(UM.userToInterest.interestToAuthor.authorTo
Book.bname)
      endIf
endWhen
```

in SeRQL the query is as follows:

```
SELECT BN
FROM
    {UM:User}userToInterest{}degree{D};
     interestToAuthor{}authorToBook{B}bname{BN}
WHERE
    D > 100 AND B IN SELECT * from session:books
```

This simple query filters books from given selection (stored in session variable session:books and retrieved by the inner sub-query) that are written by authors in which the user degree of interest is greater than 100.

5. Deploying PRML rules under rule engines

Rule engines (RE from now on) provide a comprehensive set of tools for developing, deploying and managing rules (resolve conflicts…). They provide for clear separation of rules from the application code. We claim that this is a good strategy to abstract our personalization specification because we keep the personalization orthogonal to the prime functionality of the application. The idea of developing a RE perfectly adapted to the whole system can seem an attractive option, but the down side of this choice is the time needed to develop it. Moreover, efficient REs rely on extremely complex expertise and algorithms. To reduce efforts and risks the most convenient option would be to use an existing RE like [1] [10] to support personalization rules. The only requisite for using them is to use the rule language they process.

We claim that PRML rules can be easily translated into the rule representation for commercial REs using XSLT scripts. Every web design method with a proprietary rule language would have to map its language to the rule language of the RE. Specifying the personalization with PRML also is an advantage in that sense.

For our purposes we have chosen the Simple Rule Mark-up Language (SRML) [16], an XML-based rule language format initiated by the ILog software vendor company. This language can easily be executed on any conforming RE. In a preliminary attempt to do a mapping to SRML we have achieved the following conclusions:

- *Events* can be mapped to SRML only regarding event parameters. Rules are identified by their names that are transferred to SRML. The proper rule to be triggered when an event arises is selected by the identifier of the rule's event in SRML that matches the identifier in an executable script. The script invokes the rule engine method passing in the rule name and event parameters.

- *PRML Conditions* can be mapped to SRML conditions which are composed of test expressions, and can be simple conditions or not conditions. *Simple conditions* can be bound to variables while *not conditions* cannot.
- *PRML Actions* for updating the content (*SetContent* statement) can be easily mapped to the *modify* statement in SRML. The mapping to SRML of the actions for updating the presentation and navigation in PRML (filtering, sorting) are still under consideration.

As an example to map to SRML a content update we consider the following PRML rule:

```
When navigation.Recommendations(NM.Book book) do
If (UM.userToInterest.degree>100) then
SetContent(UM.userToInterest.degree,10)
```

This rule can be mapped to the following SRML code:

```
<rule name="NRecommendations">
<conditionPart>
  <simpleCondition className="Interest"
objectVariable="i">
     <binaryExp operator="gt">
       <field name="degree"/>
       <constant type="integer"
        value="100"/>
       </binaryExp>
   </simpleCondition>
 </conditionPart>
 <actionPart>
 <modify>
     <variable name="i"/>
       <assignment>
         <field name="degree"/>
         <constant type="integer" value="10"/>
        </assignment>
 </modify>
  </actionPart>
   </rule>
```

Mapping from PRML to SRML (or to other languages) is still a subject of research and has still to be improved and studied deeply.

6. Conclusions

In this paper we offer a general solution to the lack of reusability of personalization specifications. This solution is based on a high level rule language called PRML (Personalization Rules Modeling Language) to specify at design time the personalization. This specification can be mapped (implemented) to different existing web design approaches. The specification is independent from any other application functional specification, so you can re-use it for different (development) environments. Future directions include the implementation of PRML in context of other web design methods and situations of personalization. Also a complete study of the integration of a rule engine component with the development platform is intended.

7. References

[1] Blaze Advisor. http://www.blazesoft.com/product/advisor

[2] Brickley, D., Guha, R..: RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation 10 February 2004

[3] Casteleyn, S., De Troyer, O., Brockmans, S.: "Design Time Support for Adaptive Behaviour in Web Sites", In Proceedings of the 18th ACM Symposium on Applied Computing, ISBN 1-58113-624-2, Melbourne, USA (2003), pp. 1222 – 1228.

[4] Casteleyn S., Garrigós I., De Troyer O.: "Automatic Runtime Validation and Correction of the Navigational Design of Web Sites", In Web Technologies Research and Development - APWeb 2005. Springer-Verlag, ISBN 3-540-25207-X, LNCS 3399. Shangai, China (2005), pp 453-463.

[5] Dayal U.: "Active Database Management Systems", In Proc. 3rd Int. Conf on Data and Knowledge Bases, pp 150–169, 1988

[6] Facca F. M., Ceri S., Armani J. and Demaldé V., Building Reactive Web Applications. Poster at WWW2005, Chiba, Japan (2005).

[7] Garrigós, I. ,Casteleyn, S., Gómez, J.: "A Structured Approach to Personalize Websites using the OO-H Personalization Framework". In Web Technologies Research and Development - APWeb 2005. Springer-Verlag, ISBN 3-540-25207-X, LNCS 3399. Shangai, China (2005), pp 695-705.

[8] Gómez, J., Cachero, C., and Pastor, O.: Conceptual Modelling of Device-Independent Web Applications, IEEE Multimedia Special Issue on Web Engineering (2001) pp 26-39.

[9] Houben, G.J., Frasincar, F., Barna, P, and Vdovjak, R.: Modeling User Input and Hypermedia Dynamics in Hera (ed.): International Conference on Web Engineering (ICWE 2004), Lecture Notes in Computer Science, Vol. 3140, Springer-Verlag, Munich(2004) pp 60-73.

[10] ILog JRules,http://www.ilog.fr/products/jrules/features.cfm

[11] Kappel G., Retschitzegger W.,Kimmerstorfer E., Proll B, Schwinger W. & Hofer Th. "Towards a Generic Customization Model for Ubiquitous Web Applications". Proceedings of the 2nd International Workshop on Web Oriented Software Technology (IWWOST), in conjunction

with the 16th European conference on Object-Oriented Programming (ECOOP), Malaga, Spain, June 2002

[12] Koch, N.: “Software Engineering for Adaptive Hypermedia Systems, Reference Model, Modeling Techniques and Development Process”, PhD Thesis, 2001

[13] OpenRDF, The SeRQL query language, rev. 1.1, url: http://www.openrdf.org/doc/users/ch06.html

[14] Rik Gerrits, "Business Rules, Can They Be Re-used?" Business Rules Journal, Vol. 5, No. 9, URL: http://www.BRCommunity.com/a2004/b203.html, 20 04

[15] Schwabe, D. and Rossi, G. A Conference Review System with OOHDM. In First Internacional Workshop on Web-Oriented Software Technology, 05 2001.

[16] Simple Rule Markup Language, http://xml.coverpages.org/srml.html

[17] UML 2.0 OCL specification www.omg.org/docs/ptc/03-10-14.pdf

[18] Vdovjak R., Frasincar F., Houben G.J., and Barna P.: “Engineering Semantic Web Information Systems in Hera” Journal of Web Engineering (JWE), Volume 2, Number 1-2, pages 3-26, Rinton Press, 2003