



Universitat d'Alacant  
Universidad de Alicante

A-OOH: Extending Web Application Design  
with Dynamic Personalization

Irene Garrigós Fernández



Tesis

**Doctorales**

[www.eltallerdigital.com](http://www.eltallerdigital.com)

UNIVERSIDAD de ALICANTE

PhD THESIS

A-OOH: Extending Web  
Application Design with Dynamic  
Personalization

Irene Garrigós Fernández

Advisor: Jaime Gómez

Department of Software and Computing Systems  
University of Alicante  
Alicante 2008

This work has been partially supported by the ESPIA (TIN2007-67078) project from the Spanish Ministry of Education and Science, by the DADASMECA project (GV05/220) from the Valencia Ministry of Enterprise, University and Science (Spain), and by the DADS (PBC-05-012-2) project from the Castilla La Mancha Ministry of Education and Science (Spain).



## Abstract

Web idiosyncrasy introduces new challenges for the design process that go further than the specification of the navigation map, including aspects such as continuous website evolution together with the existence of a heterogeneous audience that implies that the website design should cope with different users' needs, goals, preferences, and capabilities. Adapting the information content and services for different user groups as well as for concrete users makes websites available to a broad audience and so, it has become a vital feature of modern Web applications. This adaptivity to individual users or user groups in a website is called personalization. We define Web personalization as the process of changing the content and structure of a website to adapt it to the specific needs, goals, interests and preferences of each user taking advantage of the user's navigational behaviour.

When designing Personalized Websites (PWS), ad-hoc design becomes unworkable. Website modeling methods provide a conceptual and systematic approach to complex Web application design and implementation.

The first aim of this dissertation was to add personalization to the Object Oriented Hypermedia (OO-H) approach. For this purpose, OO-H is extended becoming A-OOH (adaptive OO-H). For being able to specify personalization strategies, a high level, ECA rule based language was defined called PRML (*Personalization Rules Modeling Language*). This language allows to specify the personalization at design time as an orthogonal concern of the website, independent of the underlying technology. The double purpose of PRML is making possible the reuse of personalization strategies among different approaches and allowing the definition of complex (and specific) personalization actions. For this purpose, two conformance levels are defined in this language: PRML Lite and PRML Full. PRML Full provides the designer a way of specifying more complex personalization actions than PRML Lite limiting the reusability.

We argue that PRML Lite can be exported to the specifics of different Web methodologies. This implies that the PRML language is reusable as well as the personalization strategies defined with it. In this work it is explained how PRML Lite can be exported to the specifics of the UWE and Hera approaches defining a set of transformations to conceptually show it. A prototype tool that

automatically implements the defined transformations (*the PRML translator*) is also presented.

Another important contribution is AWAC, a prototype CAWE tool for the automatic generation of adaptive Web applications using PRML. This tool implements the A-OOH (Adaptive Object Oriented Hypermedia) methodology.



Universitat d'Alacant  
Universidad de Alicante

## Acknowledgements

First of all, I wish to express my deepest gratitude to *my family* for their support and understanding during the elaboration of this work.

I also wish to sincerely thank my supervisor, *Jaime Gómez*, for his assistance, support, advices and encouragement.

Next, I would like to thank the members of my jury and external reviewers, for their valuable advices and assistance. I want to stress my gratitude to *Sven Casteleyn*, whose interest in my work, valuable discussions and encouragement had a great influence on my research progress. I also want to give a special word of thanks to *Cristina Cachero* for her effort done organizing the defence.

I would like to thank my colleagues of my research group for their assistance and support, especially to *J.Norberto Mazón*, *Cristina Cachero* and *Santiago Meliá*.

I would also like to thank my colleagues in the department, in particular to *Fernando Llopis* for his understanding during the elaboration of the present dissertation. I also want to thank *Miguel Ángel Varó* and *Miguel Ángel Baeza* for their assistance in the performance of several experiments.

Finally I want to thank the students which research and efforts made possible the implementation of this PhD, *Cristian Cruz* and *Armando Albert*.

Last, I would like to thank all the people who contributed in some way to the elaboration of the present Phd.



# Contents

<b>Abstract.....</b>	<b>ii</b>
<b>Acknowledgements.....</b>	<b>iv</b>
<b>Contents.....</b>	<b>vi</b>
<b>Chapter 1: Introduction.....</b>	<b>1</b>
1.1. Motivation .....	1
1.2. Proposal .....	5
1.3. Contributions.....	8
1.4. Structure of the dissertation.....	9
<b>Chapter 2: Adaptive Hypermedia and Adaptive Web-based Systems.....</b>	<b>13</b>
2.1. Hypertext and Hypermedia.....	13
2.1.1. Definitions.....	13
2.1.2. The Dexter Hypertext Reference Model.....	15
2.1.3. Extensions to the Dexter Reference Model.....	17
2.2. Adaptive Hypermedia.....	19
2.2.1. Types of AHS.....	19
2.2.2. According to which information Adapt?.....	21
2.2.3. Methods and Techniques.....	22
2.2.4. Application Areas of AHS.....	23
2.2.5. AHAM (Adaptive Hypermedia Application Model).....	24
2.3. Adaptive Web-based Systems.....	28
2.3.1. OOHDM.....	28
2.3.2. WebML.....	31
2.3.3. WSDM.....	34
2.3.4. UWE: UML-based Web Engineering .....	39
2.3.5. Hera.....	42
2.3.6. Other Approaches.....	46
2.4. Concluding Remarks.....	47
<b>Chapter 3: The Adaptive OO-H Method.....</b>	<b>51</b>
3.1. Design Process.....	52
3.2. Requirements.....	55

3.2.1. Use Case Diagram.....	55
3.2.1.1. Example.....	56
3.3. Analysis and Design.....	56
3.3.1. Domain Analysis and Design.....	56
3.3.1.1. Example.....	58
3.3.2. Navigation Design.....	58
3.3.2.1. Navigation Access Diagram.....	59
3.3.2.2. NAD Metamodel.....	63
3.3.2.3. UML Profile for the NAD.....	67
3.3.2.4. Example.....	68
3.3.3. Presentation Design.....	69
3.3.3.1. Design Presentation Diagram.....	69
3.3.3.2. DPD Metamodel.....	74
3.3.3.3. UML Profile for the DPD.....	77
3.3.3.4. Example.....	79
3.3.4. Adaptation Design.....	82
3.3.4.1. User Model.....	82
3.3.4.2. UM Metamodel.....	86
3.3.4.3. UM Profile.....	89
3.3.4.4. UM Example.....	90
3.3.4.5. Personalization Model.....	91
3.4. Implementation.....	94
3.5. Test.....	95
3.6. Conclusions.....	95
<b>Chapter 4: PRML: Personalization Rules Modeling Language.....</b>	<b>97</b>
4.1. Requirements for PRML Lite.....	100
4.2. Case Study: an Online Library.....	102
4.3. PRML Lite Fundamentals.....	107
4.3.1. Timing of the Adaptation.....	107
4.3.1.1. Events: PRML Lite Specification.....	108
4.3.2. Conditions in PRML Rules.....	110
4.3.2.1. Reference to the Models in PRML Conditions.....	110
4.3.2.2. Conditions: PRML Specification.....	111
4.3.3. Actions Supported.....	112
4.3.3.1. Updating a Value (SetContent).....	114
4.3.3.2. Filtering Attributes in the NM Nodes (selectAttribute).....	115

4.3.3.3. Filtering Node Instances (selectInstance).....	116
4.3.3.4. Link Hiding (hideLink).....	116
4.3.4. Rules Features.....	117
4.3.5. Execution Order.....	118
4.4. Conclusions.....	119
<b>Chapter 5: PRML Full.....</b>	<b>121</b>
5.1. Extension of the Case Study.....	121
5.2. PRML Full Fundamentals.....	125
5.2.1. PRML Full: Timing of the Adaptation.....	127
5.2.1.1. Events: PRML Full Specification.....	128
5.2.1.2. Support for Complex Browsing Behaviour in the Case Study.....	131
5.2.2. Actions Supported.....	133
5.2.2.1. Sorting Node Instances (sort).....	134
5.2.2.2. Adding Links (addLink) and Adding Filters (addFilterToLink).....	135
5.2.2.3. Deleting Links (deleteLink).....	137
5.2.2.4. Grouping Users.....	137
5.2.2.5. Support for Behaviour Patterns.....	143
5.2.3. Designer Guidelines.....	147
5.3. Conclusions.....	148
<b>Chapter 6: Automatic Generation of Adaptive Websites.....</b>	<b>149</b>
6.1. Related Work.....	150
6.2. AWAC: Adaptive Web Applications Creator.....	151
6.2.1. Generated AWAC Application Architecture.....	153
6.2.2. PRML Manager: Personalization Management at Runtime.....	154
6.2.3. AWAC: PRML Support.....	157
6.3. Experiment Setup.....	159
6.3.1. Step 1: Creating the A-OOH Models.....	159
6.3.2. Step 2: Adding Personalization Using a PRML File.....	165
6.3.3. Step 3: Generating the Web Application with AWAC....	169
6.4. Rules management at runtime.....	173
6.5. Results.....	176
6.6. Conclusions.....	178

<b>Chapter 7: PRML: Portability to the UWE approach.....</b>	<b>181</b>
7.1. Query / View / Transformation Language.....	182
7.2. Transformations from PRML to the Specifics of UWE.....	183
7.2.1. The UWE Adaptation metamodel.....	184
7.2.2. Sequence of QVT Relations.....	185
7.2.2.1. PRMLRule Relation.....	187
7.2.2.2. Events Relations.....	188
7.2.2.2.1. PRMLNavigation Relation.....	188
7.2.2.2.2. PRMLLoadElement Relation.....	188
7.2.2.2.3. PRMLSessionStart Relation.....	189
7.2.2.2.4. PRMLSessionEnd Relation.....	190
7.2.2.3. PRMLAction Relation.....	190
7.2.2.4. Actions Relations.....	191
7.2.2.4.1. PRMLSelectInstance Relation.....	191
7.2.2.4.2. PRMLSelectAttribute Relation.....	192
7.2.2.4.3. PRMLHideLink Relation.....	194
7.2.2.4.4. PRMLSetcontent Relation.....	196
7.2.2.5. Foreach and Condition Relations.....	197
7.2.2.5.1. PRMLForEach Relation.....	197
7.2.2.5.2. PRMLForeachSC Relation.....	198
7.2.2.5.3. PRMLCondition Relation.....	198
7.2.2.5.4. PRMLConditionPre Relation.....	199
7.2.2.5.5. PRMLConditionPreForAll Relation.....	200
7.2.3. Process transformation example.....	201
7.3. PRML To UWE: Transformation Examples.....	203
7.4. Conclusions.....	206
<b>Chapter 8: PRML: Portability to the Hera approach.....</b>	<b>209</b>
8.1. Transformations from PRML to the Specifics of Hera.....	209
8.1.1. The HERA Adaptation metamodel.....	210
8.1.2. Sequence of QVT Relations.....	211
8.1.2.1. PRMLRule Relation.....	213
8.1.2.2. Events Relations.....	213
8.1.2.2.1. PRMLNavigation Relation.....	214
8.1.2.2.2. PRMLLoadElement Relation.....	215
8.1.2.2.3. PRMLSessionStart Relation.....	217
8.1.2.2.4. PRMLSessionEnd Relation.....	217
8.1.2.3. PRMLAction Relation.....	218

8.1.2.4. Actions Relations.....	219
8.1.2.4.1. PRMLSelectInstance Relation.....	219
8.1.2.4.2. PRMLSelectAttribute Relation.....	220
8.1.2.4.3. PRMLHideLink Relation.....	221
8.1.2.4.4. PRMLSetcontent Relation.....	222
8.1.2.5. Foreach and Condition Relations.....	223
8.1.2.5.1. PRMLForEach Relation.....	224
8.1.2.5.2. PRMLForeachSC Relation.....	224
8.1.2.5.3. PRMLCondition Relation.....	225
8.1.2.5.4. PRMLCondition2 Relation.....	226
8.1.2.5.5. PRMLConditionPreForAll Relation.....	227
8.1.3. Process transformation example.....	228
8.2. PRML To Hera: Transformation Examples.....	231
8.3. Conclusions.....	237
<b>Chapter 9: Conclusions.....</b>	<b>239</b>
9.1. Contributions.....	239
9.2. Limitations.....	241
9.3. Future Work.....	242
<b>Appendix A: BNF specification of PRML Lite.....</b>	<b>245</b>
<b>Appendix B: BNF specification of PRML Full.....</b>	<b>249</b>
<b>Appendix C: PRML Translator.....</b>	<b>253</b>
<b>Appendix D: Spanish summary.....</b>	<b>259</b>
<b>References.....</b>	<b>287</b>



Universitat d'Alacant  
Universidad de Alicante

# Chapter 1

## Introduction

This chapter introduces the content and rationale of the present dissertation. Section 1.1 explains the motivation and problem statement. Section 1.2 presents the goals of the present work. The main contributions of this work are presented in Section 1.3. Finally Section 1.4 summarizes the main structure of the dissertation.

### 1.1 Motivation

The development of the World Wide Web over the past years has sparked innovation and challenged technicians all over the world. In the beginning, the Internet was government funded. But today, it becomes more of a tool for businesses and personal users. As technology increases, and the Internet becomes more ubiquitous in everyday life, evaluating how important such a service is to individuals becomes a key issue for Internet providers.

The World Wide Web has changed the way we communicate, consult and share information. It has a big impact on our daily lives, influencing the way we work, educate, drive commerce, etc.

In the last decade, the number and complexity of websites and the amount of information they offer is rapidly growing. We face a new, wide spectrum of Web applications that leads to new challenging and demanding requirements that go further than the specification of the navigation map, including aspects such as continuous website evolution together with the existence of a heterogeneous audience that implies that the website design should cope with different users' needs, goals, preferences, and capabilities. Moreover websites typically serve large audience what can lead to maintenance and usability problems [Bevan, 2005; Nielsen, 1992].

Moreover, the WWW complexity leads to navigation and comprehension problems. Traditional Web systems were based on "one size fits all" approach in which the same information is presented in the same way to all the users.

This approach has some negative effects like the inability of satisfying the heterogeneous needs, preferences and goals of the audience. The ad-hoc development of Web-based systems lacks a systematic approach and quality control. Web Engineering, an emerging new discipline, advocates a process and a systematic approach to development of high quality Web-based systems. In this context Web Design Methodologies appeared [Casteleyn et al, 2003; Ceri et al, 2003; Gómez et al, 2001; Houben et al, 2004; Kappel et al, 2002; Koch, 2001; Rossi et al 2001], giving solutions both for designers (e.g. design support, help in determining consistent structuring, easier maintenance) and for users (e.g. better tailored content, easier navigation).

To better accommodate the individual user, the development of personalized web systems (PWS) is needed. These are systems with an ability to adapt their behaviour to the individual users' (and group of users) goals, interests, tasks, capabilities and needs. There are many definitions given for personalization [Cingil et al, 2000; Blom, 2000; Kim, 2002; Wang and Lin, 2002], however there is not a universal definition. The personalization aim is to increase the relevance of the system to an individual. Many of the given definitions are converging to the objective of delivering to a user or group of users relevant information that is retrieved, transformed, and/or deduced from information sources. Some definitions agree in taking into account not only the user preferences or context for personalizing the website, but also emotional or mental needs of the user, aroused by external influences. Web personalization is defined here as the process of changing the content and structure of a Web site to the specific needs, goals, interests and preferences of each user by taking advantage of the user's navigational behaviour and user context.

When designing Personalized Websites (PWS), ad-hoc design becomes unworkable. Many existing approaches are based on rules and offer powerful specification means and tools for personalization (e.g. ILog JRules, LikeMinds, WebSphere, Rainbow,...) that make the use of personalization policies easier. These approaches have some problems like the low abstraction level and the insufficient separation from the application's functional specification that causes reuse problems and make the maintenance and scalability of the resulting personalized applications difficult.

Regarding website modeling methods many of them support the design and implementation of PWS. Their way of modeling the actual adaptation varies among the different methodologies, some use conditions to specify design

alternatives, other use rules to specify the adaptive behaviour, other methods use queries over the data model, etc. These differences force the designer of a PWS to choose one of these approaches depending on his particular needs and preferences. This can lead to different problems for the designer:

*P1. No reusability*

A personalization strategy specified with a Web design method A cannot be reused when designing a website with a different Web design method B. This is due to several factors:

- The personalization specification is not independent of the underlying technology.
- The different design methods support different personalization actions.

*P2. Restricted Personalization: Limitations*

All the existing approaches have limitations for specifying personalization. As an example some of them do not consider adapting the presentation, and some of them do not consider adding or removing links. The designer can have problems deciding the approach to use to cope with the personalization requirements of the website.

*P3. No CAWE Tool*

Most approaches do not provide an underlying CAWE<sup>1</sup> tool for Web Engineering and even less provide a tool to support personalization modeling. The lack of such tools causes the personalization to be implemented in an ad-hoc manner. Moreover, the different (adaptive) methodologies are not properly tested.

The consequence is that the designer would only choose the approaches with a generation tool.

*P4. Complexity*

Personalization is considered a full discipline, i.e. not only related to the Web, and it is present in many different fields like television [Chorianopoulos and Spinellis, 2002], education [Apple et al, 2004], medicine [The Personalized Medicine Coalition], etc. Regarding Web Engineering the persons responsible for personalization may be

---

<sup>1</sup> Computed Aided Web Engineering

knowledgeable in psychology and sociology; they are not necessarily experienced programmers. That is why we claim that, analogous to the role of the *presentation designer*, it is necessary to define a role of a designer responsible for defining personalization in a website as an orthogonal aspect, i.e. a *personalization designer*<sup>2</sup>.

The different Web methodologies have different languages for the specification of personalization. When defining a PWS, depending on the Web methodology used, the *personalization designer* has to learn a different language for specifying personalization. This can be a cumbersome task for the designer, who, as aforementioned, may not be an experienced programmer. Moreover:

- Some of these languages are difficult to learn and use.
- These languages have too specific syntax related with the modeling elements of the approach, which obscures its understanding and comparison with other personalization languages.
- Most of these languages are not domain (personalization) specific languages. This causes that are more difficult to use for an inexperienced programmer.

*P5. Personalization: only defined at design time*

Another problem for the designer can be to add personalization to a website that is already generated. When using a Web modelling approach in which the personalization is embedded in the methodology, the designer should modify all the models to add adaptivity to the site and regenerate the web pages. When the personalization is tightly intertwined with the rest of the application and the underlying technology, some problems arise like the difficulty of maintenance of personalization and the inability to reuse the personalization specification among different approaches and a more complex authoring.

It is difficult to predict at design time who will be the users of the website and how they will use the website to define an optimal structure of the Web application. To define an effective navigation structure for the site runtime information about the usage of the website is needed. This information can be used to personalize for one particular user or a group of users.

---

<sup>2</sup> Throughout the dissertation the terms *personalization designer* and *designer* are considered equivalent.

Web designers could benefit from this information, specifying at design time which adaptation will be possible at runtime. This dissertation aims to allow design time specification of runtime personalized behaviour for websites.

## 1.2 Proposal

To tackle the aforementioned problems, we propose to deal with personalization as an orthogonal concern. Personalization is considered as a separate aspect of the design of the website, independent of the underlying technology, allowing the reuse of personalization specifications. The proposed solution is based on a high level rule language called PRML (*Personalization Rules Modeling Language*) [Garrigós et al, 2005; Garrigós and Gómez, 2006] that can be used by the designer to specify at design time the personalization to be performed at runtime. Two conformance levels are defined in this language:

- *PRML Lite*: This level comprises the basic operations supported by the most representative Web modeling methods [Casteleyn et al, 2003; Ceri et al, 2003; Gómez et al, 2001; Houben et al, 2004; Koch, 2001; Rossi et al, 2001]. It allows the independence from the Web modeling methodology and thus, the reuse. The purpose of its definition is to be able to specify a universal reusable personalization policy.
- *PRML Full*: The purpose of this level is to give support to specific personalization actions, though limiting reusability. With the constructs defined at this level, more complex personalization strategies can be specified. PRML Lite is a subset of PRML Full; therefore the particularities defined in PRML Lite are also valid for PRML Full.

This language tackles the problems described above in the following way:

### *S1. Reusability*

Reusability in PRML is possible in two different ways: on the one hand, the rule language can be reused in different approaches and on the other hand, a personalization strategy can be reused within different Web methodologies<sup>3</sup>.

---

<sup>3</sup> as a consequence of the first way of reusability

- The reuse of the rule language within different Web methodologies is possible. To achieve this goal PRML has been defined as a method-independent language. PRML was born in the context of the OO-H [Gómez et al, 2000; Gómez et al, 2001] Web design method to extend it with personalization support<sup>4</sup>. Then, it evolved to be a generic language independent of the underlying technology. To achieve this independence it is necessary to identify the common denominator of the most representative Web design methodologies. This is done by abstracting the basic modeling elements for each of these methodologies. PRML has been defined using these abstract modeling elements as will be explained in Chapter 4.
- The reusability of personalization strategies among different approaches is also possible. For doing this, it is needed that the different Web design methods define their models according to the requirements that will be specified in Chapter 4. These requirements define the abstract models, which are Domain Model, User Model and Navigation Model which PRML can work/act on. To be able to use PRML and then reuse the same personalization strategy, these approaches must have equivalent models (same expressiveness) to specify a Web application. In this dissertation, UWE and Hera approaches are used to demonstrate this fact. Chapters 7 and 8 will present formal transformations from PRML to UWE and Hera using the standard language QVT [Object Management Group: MOF 2.0 Query/View/Transformation] to clearly show the portability of the PRML approach and its reusability among different methodologies. To illustrate this fact, concrete examples of PRML rules are transformed into Hera and UWE. The Appendix C presents a tool that automatically performs such transformations.

## S2. Advanced Personalization

As described before, it is possible to specify more complex personalization with PRML Full<sup>5</sup>. In this way the methodologies supporting the specification of richer personalization policies (than the

---

<sup>4</sup> OO-H is extended with personalization support becoming A-OOH (Adaptive OOH)

<sup>5</sup> Limiting the reusability

ones specified in PRML Lite) can also benefit from the use of PRML. PRML Full concretely allows the following:

- *Tracking complex browsing behaviour:* Besides supporting personalization on basis of (simple) user's behaviour (i.e. user's click on a link), we think it is important to be able to support the detection of more complex actions of the user (i.e. a sequence of link activation). The consequence is that the complexity of the personalization strategies we can define (based on user browsing behaviour) increases.
- *Adapting the Website for a (dynamic) group of users.*
- *More complex personalization actions.*

### S3. CAWE Tool

In our proposal we give support to PRML with AWAC (*Adaptive Web Applications Creator*), a prototype CAWE tool for the automatic generation of adaptive Web applications based on the A-OOH (Adaptive OO-H) methodology. As already mentioned, A-OOH is an extension of the OO-H approach to support the modeling of personalized Websites. A-OOH allows modeling the content, structure, presentation and personalization of a Web Application. The AWAC tool takes the A-OOH design models to generate as input to generate the (adaptive) Website. Once generated, the adaptive Website also contains three modules for managing the personalization which, at runtime, analyze the user browsing events and adapt the Website according to the personalization rule(s) which are triggered by these events. These personalization rules are specified independently and can thus be updated without modifying the rest of the application logic.

### S4. Usability

PRML is a high-level, simple and efficient language. It is a Domain Specific Language [Mernik et al, 2005] so the rules defined with it are very intuitive and easy to learn. DSLs allow solutions to be expressed in the idiom and at the level of abstraction of the problem domain. Consequently, domain experts themselves<sup>6</sup> can understand, validate, modify, and even develop DSL programs [Van Deursen et al, 2000]. As

---

<sup>6</sup> PRML users are supposed to be experts in the personalization domain.

mentioned in the Conclusions Chapter, as future work we plan to do some tests with groups of users to check the usability of the language.

The learning curve of the *personalization designer* is reduced due to two reasons. The first one is that we abstract the designer from the programming code and the second one is that the designer can define personalization using PRML. Transformations into the specifics of each methodology can (automatically) be performed. Using PRML Lite we assure that a personalization specification can be mapped to any of the existing approaches. Specific mappings for PRML Full can also be defined, yet in this case reusability among methodologies can no longer be assured.

#### S5. *Personalization: defined at design-time and at runtime*

The main goal of PRML is providing the designer with a way of specifying personalization strategies at design-time (that will be executed at runtime).

As personalization has become a key aspect in Web development, designers are often forced to add personalization to an existing website (at runtime). Due to the separation of concerns, PRML also allows the designer to add personalization to an already generated website<sup>7</sup>. The specification of the personalization is an independent module that can be integrated with the rest of the models without modifying them.

PRML's independence from the rest of the application logic is achieved by using ECA [Ying et al, 2007; Heimrich and Specht, 2003; Chakravarthy, Le, and Dasari, 1999; Dayal, 1988] rules. These rules represent a framework that allows rule triggering based on events that can be associated with particular actions that are taken by users when they are browsing a personalized Web application. Then, ECA rules are a convincing way of describing personalization policies for two reasons. First of all, they can be separated well from the other functional specification of a Web application by means of independent event triggering and secondly, they represent a straightforward implementation of well-defined personalization policies.

---

<sup>7</sup> This website must be modeled with one of the supported methodologies.

### 1.3 Contributions

The main contributions of this work are the following:

- OO-H has been extended to support personalization becoming A-OOH (Adaptive OO-H, Chapter 3). The main differences with respect to OO-H are:
  - Adaptive hypermedia systems are complex systems which require an appropriate software engineering process for their development. This is why in A-OOH the design process is based on the Unified Process (UP) and not in the spiral model as OO-H design process was based.
  - The Navigational Model has been modified separating the presentation features that were mixed in the Navigational Model of OO-H. Moreover a UML profile has been defined for using UML notation for representing the Navigational Model.
  - A Presentation Model has been added. This model also uses UML notation.
  - A User and Personalization Model have been added for being able of modeling adaptive Web applications.
  - Several UML profiles have been defined for the different A-OOH models.
- A high level, ECA rule based language to specify personalization is defined. This language can be used by the designer to specify at design time the personalization to be performed at runtime (Chapters 4 and 5).
- A detection mechanism of complex user browsing behaviour for the specification of more advanced personalization strategies (defined in Chapter 5).
- A prototype tool for the automatic generation and management of adaptive Web applications based on the A-OOH methodology has been implemented (Chapter 6).
- The prototype tool also includes a module for runtime management of rules (Chapter 6).
- Reusability of personalization strategies using PRML Lite. Transformations from PRML to the specifics for specifying personalization in different Web modeling approaches are studied (Chapters 7 and 8).
- A tool for automatically executing those transformations has been implemented (Appendix C)

## 1.4 Structure of the dissertation

The dissertation is structured as follows:

*Chapter 2* presents the background of the research described in this dissertation. It presents the concepts of hypertext and hypermedia presenting the different models used for hypertext, specially the well-known Dexter model. Moreover it focuses on adaptive hypermedia systems introducing the main concepts relating to adaptive hypermedia, adaptation methods and techniques. It also elaborates on different Web design methods that support adaptation.

*Chapter 3* describes the A-OOH method. First the design process proposed by A-OOH is presented. Afterwards the different models of the A-OOH approach are explained. Finally the conclusions of the chapter are presented and the main extensions done to the OO-H method are summarized.

*Chapter 4* explains the fundamentals of PRML Lite. The chapter begins presenting the case study used along the dissertation. PRML Lite is explained in terms of the events supported, the way of defining conditions over the data of the User Model, and the adaptation actions supported. Moreover is explained how to specify the execution order of the different PRML rules.

*Chapter 5* presents the basics of PRML Full extending the case study used in previous chapter with more complex personalization requirements. The fundamentals of PRML Full are explained, presenting its metamodel and explaining the support of complex user browsing behaviour and the different adaptation actions supported by the language. Moreover some guidelines are given to the designer for properly defining adaptivity rules with PRML Full.

*Chapter 6* presents AWAC, a prototype CAWE tool for the automatic generation of adaptive Web applications based on the A-OOH methodology. The AWAC tool takes the A-OOH design models of the adaptive Website to generate as an input. Once generated, the adaptive Website also contains three modules for managing the personalization which, at runtime, analyze the user browsing events and adapt the Website according to the personalization rule(s) triggered.

*Chapter 7* and *8* describe how PRML can be exported to the UWE and Hera design methods respectively. The implementation of these transformations is presented in the *Appendix C* of the dissertation.

*Chapter 9* presents the conclusions of the dissertation. A summary is provided, explaining the contributions and limitations of the work. Finally, possible extensions and future work are discussed.



Universitat d'Alacant  
Universidad de Alicante



Universitat d'Alacant  
Universidad de Alicante

## Chapter 2

### From Adaptive Hypermedia to the Adaptive Web

This chapter focuses on presenting the background of the research described in this dissertation. Section 2.1 presents the concepts of hypertext and hypermedia presenting the different models used for hypertext, specially the well-known Dexter model. Section 2.2 focuses on adaptive hypermedia systems introducing the main concepts relating to adaptive hypermedia, adaptation methods and techniques. Section 2.3 focuses on the World Wide Web elaborating on different Web design methods that support adaptation. Finally Section 2.4 summarizes the research background and gives some concluding remarks.

#### 2.1 Hypertext and Hypermedia

##### 2.1.1 Definitions

*Hypertext* is the concept of interrelating information elements (linking pieces of information) and using these links to access related pieces of information. (An information element or node can range from a single idea or chunk to an entire document.) A hypertext is a collection or Web of interrelated or linked nodes. A hypertext system allows an author to create the nodes and the links among them, and allows a reader to traverse these links, i.e., to navigate from one node to another using these links. Typically hypertext systems mark link access points or link anchors in some manner within a node when displaying it on a computer screen (e.g., underlined text displayed within documents on World Wide Web browsers). When the user selects the link marker, e.g., by clicking on it with a mouse cursor, the hypertext system traverses to and displays the node at the other end of the link. If a single link marker represents multiple links, the hypertext system may present the user with a list of available links. (System designers may have to rank, filter or layer this list if the number of possible links might overwhelm the reader). Hypertext user interface design principles recommend that authors label the link marker if the link's purpose or destination is not clear. Hypertext systems include many

navigation, annotation and structural features, which take advantage of the node and link structure to support authors and readers.

Many people consider the terms *hypertext* and *hypermedia* synonymous. Nominally *hypertext* refers to relating textual elements, while *hypermedia* encompasses relationships among elements of any media type. One of the existing definitions for hypertext and hypermedia are the following:

- **Hypertext:** “A term coined by Ted Nelson around 1965 for a collection of documents (or "nodes") containing cross-references or "links" which, with the aid of an interactive browser program, allow the reader to move easily from one document to another.”
- **Hypermedia:** “The extension of hypertext to include other media - sound, graphics, and video - has been termed "hypermedia", but is usually just called "hypertext", especially since the advent of the World-Wide Web and HTML”.

A hypertext system is a complex piece of software, consisting of several parts which serve a very different purpose. Campbell and Goodman proposed a division of a hypertext application [Campbell and Goodman, 1988] in the following way: The presentation level or user interface, the Hypertext Abstract Machine (serving nodes and links), the Database level (providing efficient storage and network access).

Such a separation of concerns forms the basis of all reference models for hypertext. A reference model for hypertext systems describes the possible conceptual elements and the functionality of hypertext systems, in an abstract (implementation independent) way. There are several reference models for hypertext systems:

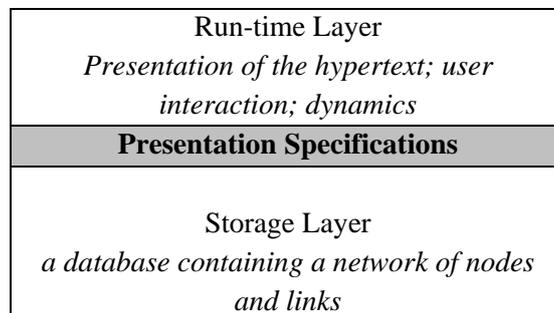
- The HAM (Hypertext Abstract Machine) model [Campbell and Goodman, 1988].
- The Trellis model [Furuta and Stotts, 1990]
- The Dexter reference model [Halasz and Schwartz, 1990]
- A formal Model written in the specification language VDM [Lange, 1990]
- The Tower Model, a more general object-oriented model [De Bra et al, 1992]

Due to its importance in the field of hypermedia being undoubtedly the most frequently used and widely referenced hypertext reference model, in the next subsection the Dexter Hypertext Reference model is described in detail.

### 2.1.2 The Dexter Hypertext Reference Model

Dexter is a well-known reference model for hypermedia systems that has been used as a standard for many years, and has been formalized and even implemented. It is an attempt to capture, both formally and informally, the important abstractions found in a wide range of existing and future hypertext systems. The goal of the model is to provide a principled basis for comparing systems as well as for developing interchange and interoperability standards. The model has been formally specified in the language Z [Spivey, 1992]. Later an Object-Z description of the Dexter model was given [Vann Ossengruggen, 1995] which is more precise, straightforward and compact.

The Dexter model is divided into three layers as we can see in figure 2-1: Runtime layer, Storage layer and Within-Component layer. An important aspect of the model is the definition of interfaces between these layers, namely the *presentation specifications* between the run-time layer and the storage layer and the *anchoring interface* between the storage layer and the within-component layer. The focus of the model is on the storage layer, which models the node/link network of the hypermedia system. The Within-Component layer is introduced to isolate the other layers from all data and media-specific details. The elaboration of the Within-Component layer is not elaborated in the Dexter Reference model. The two remaining layers are described next.



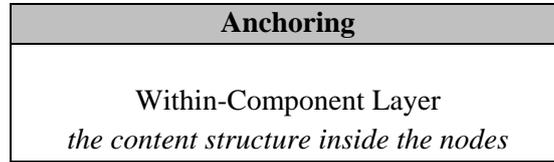


Figure 2-1: The Dexter Hypertext Reference Model

*Storage Layer:* This layer focuses on the underlying data structure used to model hyperlinks. It describes the structure of a hypertext as a finite set of components together with two functions, a resolver and an accessor function. The accessor and resolver functions are jointly responsible for “retrieving” components (i.e. mapping specification of components into the components themselves).

The notion of component replaces the weakly defined concept of nodes in the Dexter model. Components are the basic data-objects provided in the storage layer and interconnected by relational links. Component is either an atom, a link or a composite entity made up from other components. The storage layer does not attempt to model the structure within components. Components are treated as generic data containers. A special type of component is the link component. Because of the generic definition of components, the Dexter model supports computed as well as static links. Links are entities that represent relations between other components. Links can be single or bi-directional as well as multiheaded. In addition, links can be endpoints of links. Typing is supported through attributes that are added to the link component. To address specific locations within a component (which are not defined in the Dexter model) the link component relies on the anchoring interface between storage layer and the within-component layer.

In addition to the data model, Dexter also specifies a set of operations on this data model. The resolver function is responsible for resolving a component's unique ID given a component specification. This indirect addressing is used to cope with dynamic changes of component IDs, i.e., when components are edited. In the simplest case the component specification is the component ID, in which case the resolver function is the identity function. The component's ID is then fed to the component accessor function, which is responsible for

mapping that ID into the data object assigned that ID. Furthermore, there are functions for determining the interconnectivity of the network by *LinksTo* and *LinksToAnchor* operations. The first resolves all links to a given component whereas the latter resolves all links referring to a given anchor within a component

*Anchoring* provides a mechanism for addressing specific locations (span-to-span links) within all component types known to the system. Anchors consist of two parts: an anchor identifier, which is unique across the universe of discourse (not just within the scope of one hypertext), and an anchor value, which defines the actual region of the anchor within the component.

*Run-time Layer*: This layer stressed the importance of the unique, associative user interface and behaviour of a hypertext system. It captures the dynamic aspects of hypertext systems. These include especially the presentation of hypertext components and the interaction processes with the user.

The way in which a component is presented to the user is encoded in the *Presentation Specifications*. Because links are also components, link behavior can be customized as well. The same link may bring up different "modes" of the destination component depending on the status of the user who invoked the link. All link specifiers indicate a direction of the link of the form "FROM", "TO", "BIDIRECT", or "NONE". Still, the semantics of the links are not explicitly defined.

*The presentation specifications* describe the interface mechanism between the runtime layer and the storage layer. It specifies how a component is to be presented to the user.

### 2.1.3 Extensions to the Dexter Reference Model

- The *Amsterdam Hypermedia Model (AHM)* [Hardman et al, 1994] tries to tackle the complex timing and presentation relationships found in multimedia presentations. In particular, the AHM extends the Dexter model by adding high-level presentation attributes and link context. The AHM follows in its multimedia aspects the CMIF model. It allows authors to specify how individual pieces of information relate to each other over a

period of time. This is a non trivial problem; dynamic and static objects fetched from distributed sources have to be synchronized. The model has to be capable of defining requirements for links, time, and global representation semantics in a hypermedia system.

- A second extension to the Dexter Hypertext Reference Model comes from *Aarhus University* in Denmark. People there are working on computer-supported cooperative work (CSCW) aspects encountered in large engineering projects [Gronbaek and Trigg, 1996]. Cooperative work raises several new requirements for hypermedia applications such as explicit communication and coordination between team workers as well as implicit coordination through shared materials. To accomplish these, new notions have to be introduced to the model such as shared databases, event notification, open access from different platforms and from different, already developed applications. The kind of computer support for cooperative work depends on the type of cooperation. The Aarhus team has identified six different modes of cooperation, ranging from separate to fully synchronous sessions. To prove their concepts for developing a cooperative hypermedia framework, the *DeVise Hypermedia Architecture (DHM)* has been implemented. Based on a distributed object system, the DHM introduces several client and server processes compliant to the Dexter model.
- Another extension of the Dexter reference model is the *AHAM* [De Bra et al, 1999], a reference model for Adaptive Hypermedia Systems (AHS). First extension of the Dexter model has been including a mechanism to store persistent information about a user in order to be able to perform adaptation. In the Dexter model information about the user is stored in the Run-time layer (session entity) but this information is not stored permanently. Therefore AHAM extends the Dexter model by including a user profile and adaptation in the Storage Layer, without throwing away the structures and functions Dexter already offers. The AHAM model is described more extensively in section 2.2.5.
- The Munich Reference Model [Koch, 2001] for Adaptive Hypermedia Applications is an extension of the Dexter Model. It is an object-oriented specification based on UML models and the Objects Constraints Language (OCL). This approach is represented by a metamodel showing all model elements and how they are related.

## 2.2 Adaptive Hypermedia

In this section we give an overview of the adaptive hypermedia (AH), describing the different types of adaptive hypermedia systems (AHS), the main features, techniques and methods. Finally we present in detail a well-known reference model for AH, the AHAM reference model.

*“By adaptive hypermedia systems we mean all hypertext and hypermedia systems which reflect some features of the user in the user model and apply this model to adapt various visible aspects of the system to the user.”* [Brusilovsky, 1996].

According to this definition given by Brusilovsky, an AHS must fulfill three requirements:

- It must be a *hypertext or hypermedia system*.
- It must include a *user model* to describe the user.
- It must provide an *adaptive mechanism* for the dynamic adaptation of the hypermedia on the basis of the state of the user model.

We should clearly differentiate between the different types of AHS depending on the way the adaptation is performed.

### 2.2.1 Types of AHS

Depending on the way of gathering the information needed for adapting a system and the cause that triggers the adaptation, we can classify the AHS into four types: *adaptable, adaptive, dynamic or proactive* [Kappel et al., 2001; Fraternali, 2000, Wu, 2002] as we can see in figure 2-2.

- We can say an AHS is *adaptable* (or that supports *adaptation*, also called *adaptability*) when the information we use to perform the adaptation is acquired in an explicit way (i.e. asking the user) and the personalization is caused by an explicit action of the user. In these systems the user is allowed to configure the system explicitly by changing some parameters and the system then adapts its behaviour accordingly. When the adaptation is performed depends on the specifications of the system (e.g. at the beginning of the session). An example of such application would be the preferences introduced by the user in a form (change of colours, fonts, reordering menus...).

- An AHS is *adaptive* (i.e. supports *adaptivity*) when it adapts autonomously monitoring the user's behaviour. This means that in this case the information needed for adapting the application is gathered in an implicit way, through the application context or tracking the activity of the user in the system [Bulterman, Rutledge, Hardman and van Ossenbruggen, 1999] and then adapts the system dynamically to the current state of the user model. The only difference with the adaptable applications is the way of acquiring the information. An example of such system could be a recommendation system used in many applications like Amazon [amazon].
- An AHS is *dynamic* when it adapts autonomously monitoring the user's behaviour but instead of changing predefined presentations as in the case of adaptive AHS, constructs a presentation from pieces of information [De Bra, 1999].
- Finally we can say that an AHS supports *proactivity* [Fraternali, 2000] when it is able to detect by itself the changes in the interaction context and reacting consequently acting as observers [Gamma, 1995] of the environment. The main difference with the adaptable and adaptive applications is that the adaptation is caused by an environment event (e.g. time interval, changes in the context...) without any action of the user and independently of how the information is gathered (either explicitly or implicitly).

<b>Acquisition of information</b> <b>Adaptation cause</b>	<i>Explicit</i>	<i>Implicit</i>
<i>User Action</i>	<b>Adaptable</b>	<b>Adaptive (if presentation generation = Dynamic)</b>
<i>Environment Event</i>	<b>Proactive</b>	

Figure 2-2: Types of AHS

*Personalization* can be defined as adaptation for a single user. The personalization of applications, defined as the ability of dynamic modifications with the goal of provide the user the impression of being working with an application specifically designed to satisfy his particular needs (also called by some authors *One to One Delivery* ).

In this dissertation we define *personalization* as the process of changing the content and structure of a website to adapt it to the specific needs, goals, interests and preferences of each user taking advantage of the user's navigational behaviour.

### 2.2.2 According to which Information Adapt?

In AHS adaptation is based on taking into account the information stored in the user model. Kobsa et al. [Kobsa et al, 2001] proposed distinguishing adaptation to user data, usage data and environment data.

- *User data* comprise the traditional adaptation target, encapsulated by several characteristics of the users. Examples include interests, preferences, goals, capabilities, user knowledge...
- *Usage data* comprise data about the user interaction with the systems.
- *Environment data* comprise aspects of the user's environment. Adaptation decisions based on environment data may depend on the user's location and the user platform (hardware , software, network bandwidth).

Brusilovsky [Brusilovsky, 1996] identified five features used by many existing AHS (knowledge, goals, background, hyperspace experience, preferences) and two newer features (interests and individual traits) (this is similar to the user data identified by Kobsa et al.):

- *Knowledge*: this appears to be the most important feature of the user for existing AHS. Almost all adaptive presentation techniques rely on user's knowledge as a source of adaptation. It is most often represented by an *overlay model* () which is based on the structural model of the subject domain.
- *Goals*: A user's goal is the most changeable user feature. Depending of the system it can be the goal of work (in application systems), a search

goal (in information retrieval systems) and a problem-solving or learning goal (in educational systems).

- *User's background* means all the information related to the user's previous experience outside the subject of the hypermedia system, which is relevant enough to be considered. This includes the experience of work in related areas, the user's profession, as well as the user's point of view and perspective.
- *User's experience* in the given hyperspace is how familiar the user is with the structure of the hyperspace and how easy can the user navigate in it.
- *Preferences*: For different reasons the user can prefer some nodes and links over others and some parts of a page over others. These preferences can be absolute or relative, i.e., dependent from the current node, goal and current context in general. Unlike other user model components, the preferences cannot be deduced by the system.
- *User interests*: it was not used in early AHS. This situation has changed with the rise of Web IR hypermedia systems as well as in recommendation systems.
- *User's Individual Traits*: this feature represents user features that together define the user as an individual. Examples can be cognitive factors, personality factors or learning styles. They are usually extracted from psychological tests. Researchers in general agree on the importance of modelling and using individual traits but it is not clear which features can and should be used or how to use them.

Brusilovsky also proposes adaptation to user's environment in [Brusilovsky, 2001] to adapt to both the user location and user platform.

### 2.2.3 Methods and Techniques

In [Brusilovsky, 1996] Brusilovsky distinguishes two distinct areas of adaptation: content level adaptation or *adaptive presentation* and link level adaptation or *adaptive navigation support*. Adaptive presentation was subdivided into text adaptation and multimedia adaptation technologies. Adaptive navigation was subdivided into link hiding, sorting, annotation, direct guidance and hypertext map adaptation.

A complete description of these techniques and methods can be found in [Brusilovsky, 2001].

#### 2.2.4 Application Areas of AHS

In 1996 [Brusilovsky, 1996] were identified six kinds of AHS – *educational hypermedia*, *on-line information systems*, *on-line help systems*, *information retrieval hypermedia*, *institutional hypermedia*, and *systems for managing personalized views in information spaces*. Educational hypermedia and on-line information systems are now established leaders. Information retrieval (IR) hypermedia is challenging the leaders.

The most popular area for AHS is *educational hypermedia*. They are also known as instructional, teaching, tutoring, training, learning or e-learning systems. Storing the user's knowledge of the subject being taught allows offering an adaptive learning system based on the improvements of the users. They assume that the AHS will be used by a heterogeneous group of users in terms of knowledge. Some known adaptive educational hypermedia systems are ELM-ART [Brusilovsky et al, 1996a], InterBook [Brusilovsky et al, 1998], 2L670 [De Bra, 1996], TANGOW [Brusilovsky, Schwarz and Weber, 1996], SmexWeb [Albrecht, Koch and Tiller, 1999], ISIS-Tutor [Brusilovsky, 1997] and AHA [De Bra and Calvi, 1998a].

Another popular application area for AHS is *on-line information systems*. This group includes e-commerce applications, recommendation systems, digital libraries, electronic catalogues and all classes of online documentation. The goal of these systems is to provide reference access to information for the users with different knowledge level on the subject. Each node of the hyperspace represents one concept of the subject and contains several pages of information. Similar to educational hypermedia, this kind of systems have problems with satisfying the needs of very different users. Examples of on-line information systems are PUSH [Höök, 1998], Swan [Garlatti, Iksal and Kervella, 1999], MetaDoc [Boyle and Encarnacion, 1994], AVANTI [Fink, Kobsa and Nill, 1997], CiteSeer [Ballacker, Lawrence and Giles, 2000] and commercial products such as Amazon [amazon].

The *online-help systems* serve on-line information about computer applications (such a spreadsheet, programming environment or expert system) which is required to help the users this system. They are not independent as the on-line information systems but are attached to their application system. Their assistance consists of presenting help information when requested and automatically recognising when the user needs some help. The advantage of these systems is knowing the context in which the user is working. Examples of on-line help systems are ORIMUHS [Encarnaçao, 1997] and WING-MIT [Kim, 1995].

*Information retrieval (IR) hypermedia systems* is a new class of IR systems which combine traditional information retrieval techniques with a hypertext-like access from the index terms to documents and provide the possibility of browsing the hyperspace of documents using similarity links between documents [Agosti, Melucci and Crestani 1995; Helmes, Razum and Barth, 1995].

Another of the news application areas is *institutional information systems* which servers on-line all the information required to support the work of some institution. Hynecosum [Vassileva, 1996] is a hypermedia information system for hospitals. It allows adaptive navigation support for users of different level of experience. A problem for these systems is related to new employees who are not familiar with the structure of the hyperspace and can get lost even in their small professional subarea.

Finally in the classification made by Brusilovsky we find the *systems for managing personalized views in information spaces*. Examples of this type of systems are Information Islands [Waterworth, 1996] and Basar [Thomas, 1995]. Many users have to access to one or more subsets of all the hyperspace for their everyday work. To protect themselves from the complexity of the overall hyperspace, they are maybe interested in defining personalized views of the entire hyperspace. These views require permanent management: searching for new and relevant items and identifying expired or changed items. Adaptation to the user goals, interests and background can help to solve the identified problems [Thomas, 1995; Thomas and Fischer, 1996].

### 2.2.5 AHAM (Adaptive Hypermedia Application Model)

AHAM [De Bra et al, 1999] is a reference model for adaptive hypermedia systems (AHS) born in the context of *e-learning*. It is based on the Dexter reference model [Halasz and Schwartz, 1990; Halasz and Schwartz, 1994] augmenting it with features for doing adaptation.

AHAM focuses on the Storage layer and uses the anchoring and the presentation specifications as an interface to the Within-Component layer and the Run-time layer (see figure 2-3). In AHAM the Storage layer consists in three parts:

- *Domain model (DM)*: describes the information domain.
- *User model (UM)*: describes user features used in adaptation.
- *Adaptation model (AM, also called teaching model)*: describes the adaptation strategies. With the help of the UM and the DM specifies how to update the UM and how to generate adaptation. It consists of adaptation rules.

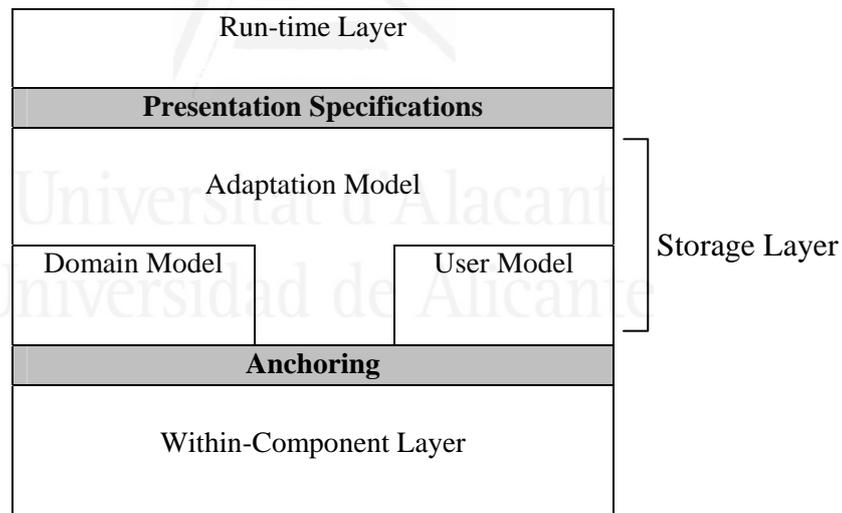


Figure 2-3: The AHAM Reference Model

**The Domain Model (DM):** The concept of component is extended to concept component, or just concept. There are three levels in the DM. At the middle level there are page components to capture the amount of information that is

presented to a user at any one time. At the highest level, for a larger amount of information, an abstract composite component (also called abstract concepts) is used. It consists of a number of pages and/or other (smaller) composite components. A page can consist of a number of smaller items that can be included or left out, but cannot be changed by the AHS. They are called atomic concepts or fragments, placed at the lowest level of the DM. The structure formed by fragments, pages and abstract concepts must form a hierarchy.

The reasons for this extension are that several adaptive presentation techniques deal with (small) parts of a node or page. In order to be able to describe these techniques at an abstract level in DM, we consider pages as composites consisting of fragments.

Another reason is that AHS often use the notion of prerequisites. To make it easy to define such prerequisites we need the ability to combine pages into larger conceptual units, which we call concepts. (Without concepts an author would have to include all the pages of a concept in a description of a prerequisite).

Instead of link components, AHAM uses concept relationships. They are defined in order to represent semantic (non-navigational) relationships between concepts (as well as navigational relationships). Concept relationships consist of a list of components. These components are given a specifier that needs to be resolved to an identifier (or a set of identifiers). In AHAM authors or system designers can define arbitrarily many types of relationships.

***The User Model (UM):*** The UM allows to maintain permanent and continuously updated information about the user. It is part of the Storage layer for being persistent across session boundaries.

AHS should be able to adapt three types of information: stable personal characteristics (i.e. interest, background...), aspects of the user's environment (including place, time, network, used device), and the relation between the user and the concepts in the domain model (i.e. interest in each concept, or knowledge about).

In AHAM they focus on the third aspect: how the user relates to the domain model when browsing the hyperspace. Therefore the UM is defined as an overlay model of the DM. For each concept in the DM, the UM stores attributes values about the concept.

AHAM does not need the presence of specific attributes in the UM. However many AHS store at least the knowledge (or knowledge\_value) which indicates how much the user knows about the concept and the read attribute indicates whether the user has to read something (a fragment, a page or a set of pages) about the concept. A less known attribute is ready (or ready\_to\_read), sometimes also referred to the desirable or recommended. It indicates wheter the user is ready to read about this concept.

The UM is represented as a table for each user with their concepts and pairs of attribute-values. Actual implementations of AHS may implement this structure in a different way. The AHA! System [De Bra and Calvi, 1997, 1998b] for instance uses a logfile (separate for every user) in which the time is logged at which a user requests a page, and also the time when the user leaves the page.

***The Adaptation Model (AM, also called Teaching Model):*** The AM defines how the user model is updated and adaptation is done by using information from the DM and the UM and the interaction of the user. These tasks are performed by an adaptation engine. AHAM uses the mechanism of language rules to express the functionality of the AM.

Adaptation can be specified in a structured way or in an ad-hoc way. Structured, or generic means that they associate a certain adaptation behaviour with concept relationships (or other structures) that exist in the domain model. In AHAM also specific rules for some given concept can be written (i.e. tied to concrete concepts and/or pages). Specific rules take precedence over the generic ones, as it is possible that a generic and specific rule contradict each other.

Rules in AHS can perform three functions: Initialization of the UM, updating of the UM and generation of the adaptation. In AHAM the user model updates

are splitted in two parts: updates done before generating the adaptation (pre), and updates done after generating the adaptation (post).

### 2.3 Adaptive Web-Based Systems

Adaptive Web-Based Systems (AWS) are a subset of AHS because the Web offers only restricted functionality of what a full hypermedia system could offer (see [Cailiau and Ashman, 1999; Casteleyn, 2005]).

Regarding the World Wide Web, the evolution from a small set of (static) linked pages towards complex Web applications has lead to user disorientation and comprehension problems, as well as development and maintenance problems for designers. In most cases, development of Web-based systems has been ad hoc, lacking systematic approach and quality control and assurance procedures. Web Engineering, an emerging new discipline, advocates a process and a systematic approach to development of high quality Web-based systems. In this context Web Design Methodologies appeared, giving solutions both for designers and for users. However, new challenges appeared, like the need of continuous evolution, or the different needs and goals of the users. Adapting the structure and the information content and services for concrete users (or for different user groups) tackles the aforementioned (navigation, comprehension and usability) problems. It has therefore become a vital feature of modern Web applications.

#### 2.3.1 OOHDM

The Object-Oriented Hypermedia Design Method (**OOHDM**) [Schwabe and Rossi, 1995; Schwabe and Rossi, 1998] and its successor, the Semantic Hypermedia Design Method (**SHDM**) [Schwabe and Moura, 2003; Schwabe et al. 2004]) allows the concise specification and implementation of hypermedia (Web) applications. This is achieved based on various models describing information (conceptual), navigation and interface aspects of these applications, and the mapping of these models into running applications, in various environments.

In OOHDM, a hypermedia application is built in a five-step process supporting an incremental or prototype process model. Each step focuses on a particular design concern, and an object-oriented model is built. Classification, aggregation and generalization/specialization are used throughout the process to enhance abstraction power and reuse opportunities. SHDM is a model-driven approach to design Web applications using five different design phases (the same as OOHDM) explained next. The difference with OOHDM is that SHDM uses semantic Web technology to represent its models.

In the *Requirements Gathering* phase, the requirements are obtained forming scenarios that are abstracted into use cases. The analysis of these use cases determines the user profiles and the tasks that the application will support. For each use case a user interaction diagram is defined, representing the interaction between the user and the application. Although the information for personalization is not explicitly extracted yet, this analysis could be extended to do so.

In the *Conceptual Design* phase a conceptual model is defined with the domain objects, relationships and required functionality. Many personalization mechanisms deal with objects and algorithms that are expressed as part of the conceptual model. OOHDM conceptual model is a class diagram while SHDM conceptual model uses semantic Web technology as RDF and OWL.

In the *Navigational Design* phase the navigation structure of the application is described in terms of navigational contexts, which are induced from navigation classes such as nodes, links, indexes and guided tours. There are two models: the navigational class model, which describes which information can be accessed and the navigational context model, which describes how this information can be reached. Nodes in OOHDM represent logical views on conceptual classes defined during domain analysis. In SHDM this views are defined using the RDF Query Language (RQL). Different navigational models may be built for the same conceptual schema to express different views on the same domain.

The *Abstract Interface Design* makes perceptible to the user the navigational structure of the application through the application interface, which is done by defining an abstract interface model. In OOHDM the abstract data view (ADV) design approach is used for describing the user interface of a hypermedia application [Cowan and Lucena, 1995]. ADVs are formal models of interface objects. In SHDM, the abstract interface model is specified using the Abstract Widget Ontology. The specification of the concrete interface design (look and feel and layout) is left to the graphic designer, since it is totally dependent on the particular hardware and software runtime environment.

Finally in the *implementation* phase, the interface objects are mapped to the implementation objects and may involve elaborated architectures, generating the actual Web site implementation code.

#### ***Adaptivity in OOHDM***

In OOHDM [Rossi et al, 2001] adaptability (i.e. adaptable applications) is supported by defining different navigational views for the same conceptual model, and of different user interfaces for the same application. In this way, the conceptual model can be customized to different user profiles and roles. Personalization is also considered in terms of link, content and context personalization. *Link personalization* consists on selecting the links that are more relevant to the user, changing the original navigation space by reducing or improving the relationships between nodes. *Content* is personalized when nodes (pages) present different information to different users. *Context personalization* is important when the same information (node) can be reached in different situations. OOHDM uses a simple declarative specification to indicate the nodes contained in a context and which user or user profiles can view a node in a particular context. *In the interface model* different layouts can be defined according to user preferences or selected devices.

To specify personalization OOHDM uses a proprietary notation with an underlying design framework. The OOHDM notation uses a small set of primitives for specifying personalized attributes and methods; more complex personalization strategies, such as using internet services, can be easily dealt

with by just applying well-known design techniques that fit naturally with the OOHDM approach. To model relevant information about the user, OOHDM adopts the user class in the Conceptual model. Attributes of the user class can subsequently be used in the Navigation Class Model to adjust the information that is shown to the user.

### 2.3.2 WebML

WebML (Web Modeling Language) [Ceri et al, 2002; Ceri et al., 2000] is a visual language for specifying the content structure of a Web application and the organization and presentation of contents in one or more hypertexts. In WebML the following models are defined:

- **Data Model:**

The design process starts with the specification of a data schema, expressing the organization of the Web application contents. The WebML Data Model adopts the Entity-Relationship (E-R) primitives or the UML class diagram.

- **Hypertext Model:**

The WebML *Hypertext Model* allows describing how contents, previously specified in the data schema, are published into the application hypertext. The overall structure of the hypertext is defined in terms of pages, units and links, organized into modularization constructs called site views and areas.

- A *site view* is a hypertext, designed to address a specific set of requirements.
- A site view is composed of *areas*, which are the main sections of the hypertext and comprise recursively other sub-areas or pages.
- *Pages* are the actual containers of information delivered to the user. They are made of *content units*, which are the elementary pieces of information extracted from the data sources by means of queries, and published within pages.
- *Content units* denote alternative ways for displaying one or more entity instances. In WebML the page content and navigation structure is captured in the (advanced) hypertext model using a predefined set of modeling primitives.
- Content units and pages are interconnected by *links* to constitute site views. With a link can be associated parameters that transfer the input values to the destination unit(s), for example for further processing by an operation unit. There are several predefined operation units, for

instance for activating external Web services or content management operations like creation, deletion, and update of entities and relations. WebML also provides units for the definition of session parameters. The whole library of units is open (new units can be defined in XML) and contains a number of data entry units for different kinds of user inputs. All contextual information passed between the units by link parameters is described in separate XML files.

Besides having a visual representation, WebML primitives are provided with an XML-based representation, which specifies additional properties, not conveniently expressible in the graphic notation. The XML specification allows the deployment of the same design into multiple rendering formats, such as HTML (standard choice), WML [Hjelm et al, 1998], SALT [SALTforum.org, 2005] or VoiceXML[W3C, 2004] for multi-model interactions. WebML is supported by a CASE tool, named WebRatio [Ceri et al, 2002] which allows the automatic generation of the application code.

#### *Adaptivity in WebML*

Differently from most conventional adaptive hypermedia systems, which mainly address the problem of adapting the results of user-generated requests, WebML stresses the importance of user-independent, context-triggered adaptation actions, which finally leads to interpret context as a “first class” actor operating independently from users on the same hypertext the users navigate.

WebML has been extended enriching the different models for adaptivity support. First of all, the data model is enriched with a context model to keep a consistent and updated representation of meta-data needed to support adaptivity. This model consists on three subschemas:

- **User profile sub-schema**, in which user, groups and site views are represented as “first-class citizens” in the application data source.
- **Personalization sub-schema**, consisting in adding personalization relationships between the entity User and some other entities to express for instance preferences of the users.
- **Context model sub-schema**, in which meta-data of the context (e.g. Device, Location...) is specified.

The hypertext model also introduces new constructs to support the specification of adaptive application behaviour.

- **Context-aware pages**, tagged with a C-label. This label indicates that some adaptivity actions are associated with the page. The assumption is that context awareness is a property to be associated only to some pages of an application. These pages are augmented with a refresh mechanism, which automatically generates refresh events at given time intervals. This allows the application to react on context-changed occurring when the user is consulting a page.
- **Adaptivity policies** express when the evaluation of adaptivity actions should be triggered, *Deferred* when user has the highest priority: adaptivity actions are evaluated after the page has been rendered according to the user's selections by automatic page requests *or Immediate Adaptivity* when context has the highest priority: adaptivity actions are evaluated when the page is accessed, prior to the actual page computation.
- **Context Clouds** represent the set of adaptivity actions attached to a page.
- **Context-Aware Containers** this construct allows to group pages (by means of site views and areas) to avoid redundancy and specify adaptivity actions to be performed for every C-page within the container.

Finally the adaptivity is specified by means of ECA rules in which the event consists on a page request, the condition for rule activation consists in the evaluation of context parameters previously acquired and the action consists in adaptations of the hypertext front-end.

In order to gather adaptivity with respect to the current state of context new mechanisms and primitives are required for:

- Specifying the acquisition of fresh context data, sensed at the client side.
- Specifying the acquisition of context data from the context model.
- Updating the Context model
- Monitoring the context model.

To evaluate conditions two control structures have been proposed (if and switch operation units).

Actions can be activated for customizing the page contents (filtering data items, including/excluding specific content units), the navigation (automatic navigation actions), the current site view (restructuring the whole hypertext) and the presentation style (by means of the Change Style unit is possible to dynamically selecting presentation style properties specified in CSS-Cascade Style Sheet- files).

Recently, in a workshop presentation of the ICWE conference [Daniel et al, 2006] it is described the integration of WebML with a language for expressing ECA rules supported by an engine for rule execution: **Chimera-Exception Language**.

Chimera-Web (new version of Chimera-Exception addressing Web events and Web adaptivity actions) is used to specify sparse adaptivity rules, whose execution is completely detached from the execution of the application itself and whose effects are not necessarily bound to instances of modelling constructs.

Such integration leads to a framework that keeps the advantages of conceptual modelling and automatic code generation. Authors claim that the resulting architecture enhances separation of concerns, and supports flexibility and evolvability. However the integration is not yet in a mature stage.

### 2.3.3 WSDM

WSDM was initiated in 1998 by De Troyer and Leune [De Troyer and Leune, 1998] and is one of the first Web site design methods. It facilitates the development of Web sites and Web applications in a systematic way. WSDM has several important pillars, which distinguishes it from other Web design methods:

- *Audience Driven*: WSDM takes as a starting point the different users with their different requirements and goals. The audience of the site is categorized into a hierarchy according these requirements, and the navigation structure of the site is based upon this hierarchy.
- *Methodology*: more than other design methods, WSDM is a methodology. Next to the necessary models and primitives to model, at different levels

of abstraction, the different concerns of a Web application, WSDM also provides the designer with a clear method on how to construct these design models.

- *Conceptual design*: WSDM makes a distinction between the conceptual design (task analysis, modeling data and functionality, and the conceptual navigation schema) and the implementation design (grouping information onto pages, layout and presentation).
- *Implementation design*: WSDM makes a distinction between the implementation design (grouping into pages, layout and presentation) and the actual implementation (e.g. java servlets, php code, html, XML, ...)

Figure 2-4 shows an overview of the different phases of WSDM, and the corresponding design models. A more detailed overview of all the phases of WSDM can be found in [De Troyer and Casteleyn, 2004; De Troyer and Casteleyn, 2001].

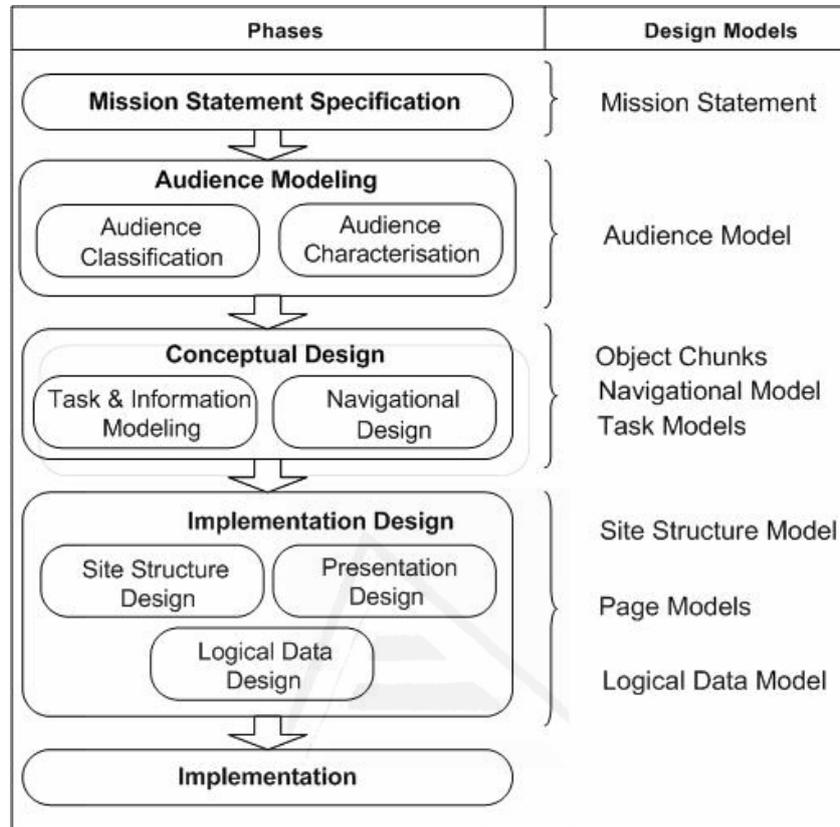


Figure 2-4: WSDM Overview

In the first phase of WSDM, the **mission statement** of the Website, we express the purpose and the subject of the Web site and declare the target audience.

Based on the mission statement, an **audience modeling phase** is performed, consisting of an audience classification and an audience characterization. During audience classification, we identify the different audience classes based upon the informational and functional requirements of the potential visitors, and classify them into an audience class hierarchy. During audience characterization, we describe the relevant characteristics for the different audience classes.

The conceptual design phase consists of two subphases: task and information modeling and navigational design. During task and information modeling, the user requirements, which were informally specified in the previous phase, are elaborated. For each requirement, a task model is specified. WSDM uses the

Concurrent Task Tree technique [Paterno et al, 1997] for its task modeling because it allows not only the hierarchical decomposition of a task in subtasks but also to specify temporal relationships between different subtasks. Next, for each elementary task, an object chunk is created. Such an object chunk is a tiny conceptual schema to formally describes the data or functionality needed by a particular elementary task. Initially, WSDM used an extended version of ORM [Halpin, 1995] to model object chunks, but currently OWL is used.

During navigational design, the conceptual navigation structure of the Website is built. A navigational model consists of nodes, which group information and/or functionality (elementary tasks) that logically belong together. Links between those nodes specify navigation paths. WSDM provides a systematic method for obtaining the navigation structure of a Website: first, the audience class hierarchy is one-to-one mapped onto a navigational model, creating a navigation track for each audience class. Next, each navigation track is further refined using the information in the task models. Nodes are created for elementary tasks; navigation links are used to reflect the temporal relationships between tasks and the relevant object chunks are connected to the nodes.

The goal of the implementation design is to provide, for the conceptual design, the necessary implementation details. The implementation design consists of three subphases, which are discussed in detail below.

The subject of the first subphase, the *Site Structure Design*, is to decide how the nodes, defined in the navigation model, will be grouped into pages. Note that in this phase we define abstract pages; each abstract page possibly gives rise to multiple concrete page instances in the actual implementation. If necessary, different Site Structures can be specified, reflecting for example different devices, contexts or platforms. The goal of the next sub phase, the *Presentation Design*, is to describe the layout (i.e., positioning and style) for all pages. WSDM provides three different sets of modeling concepts to describe the layout of a page. Each set provides a different level of abstraction.

During the *Template and Style design*, the designer specifies templates that will be used for different types of pages. Templates define the general layout of a page. To specify the Website style (fonts, colours, graphics, etc), WSDM currently uses Style Sheets. In the Page Design sub phase the designer

describes where the information on a page (defined in the object chunks) should be positioned and how it should be laid out. It is also decided how and where the links (defined in the navigational design) should be presented. A template (defined in the previous subphase) is taken as a starting point for a page.

In the final phase, the actual implementation, all the information collected so far is taken as input for the Website generation process. WSDM describes two implementation architectures: an XML-based transformation pipeline, capable of generating static Web sites, and a Java-based server-side implementation also capable of generating dynamic Web sites.

WSDM also has support for several other design concerns: localization, semantic annotations, accessibility and adaptation.

### ***Adaptation in WSDM***

WSDM was extended [Casteleyn, 2005] to support design time specification of runtime adaptation based on Web site usage information. Note that personalization is not supported (i.e. adaptivity for a single user), however the aim is to provide adaptivity for *all users*, i.e. also called optimization.

Adaptation in WSDM is supported by means of the Adaptation Specification Language (ASL), a dedicated high-level specification language. Using ASL, the designer can specify adaptation policies (*when* certain adaptation needs to be performed), and adaptation strategies (*which* adaptation needs to be performed). Adaptation strategies are specified using ASL's rule-like constructs. ASL has an event-based triggering mechanism: events (e.g., the user clicks a link, visits a page, a time interval has elapsed) will trigger the adaptation strategies (according to the specification in the adaptation policy).

ASL rules are defined to act upon the various WSDM design models. The most primitive rules are the native operations which allow to add/delete the links, nodes and pages of the models, chunks can be connected or disconnected to/from nodes, and nodes can be added or removed to/from pages. More complex ASL rules allow iteration, conditional execution of an action, declaring/using (tracking) variables or perform pre-defined operations. Finally, ASL supports expressions (mainly to specify threshold or perform statistical calculations) and set expressions (to calculate relevant subsets of design model elements).

ASL as it is shortly described above is founded on a formal representation of WSDM, with the necessary extensions to support adaptation. More in particular, based on the formalization of the different WSDM design models, a Web Site Overlay Model is defined to support the storage (at runtime) of Web site usage information. Next to that, WSDM defines operations to populate the Overlay Model and to perform basic model transformations. The model transformation operations allow manipulation of the relevant Web site design models to allow for adaptation, and correspond to the native operations in ASL.

Using ASL, several useful adaptation strategies have been specified (e.g. [Casteleyn et al, 2005]): promotion and demotion of nodes, re-ordering sequential information, validation of an (audience driven) navigation structure. Note again that the focus of adaptation in WSDM is optimizing the navigation structure for all users, not for one specific user. However, although personalization is not explicitly supported, it would only require some minor modification to include a user model (in an exactly similar way as the existing Website overlay model) and access from ASL to this user model. In the context of this dissertation, ASL is thus as relevant as any other adaptation specification mechanism which is aimed at personalization. A detailed specification of the WSDM formalization, the extensions to allow adaptation and some example adaptation strategies can be found in [Casteleyn, 2005; Casteleyn et al, 2005].

#### 2.3.4 UWE: UML-based Web Engineering

The UWE approach [Koch, 2000; Koch et al., 2001] is an object oriented approach which has as a distinguishing feature its Unified Modelling Language [UML, 1999] compliance since UWE is defined in the form of a UML profile and an extension of the UML meta-model. UWE follows the principles of the Unified Software Development Process [Jacobson et al, 1999] and supports the systematic development of Web applications focusing on the specification of adaptive (personalized) applications. An extension of ArgoUML known as ArgoUWE gives support to the approach. The fundamentals of this approach are a standard notation (UML through all the models), the precise definition of the method and the specification of constraints (with the OCL language) to increase the precision of the models.

The separate modelling of Web application concerns is a main feature of this approach. Thus, different models are built for each point of view: the content, the navigation structure, the business processes and the presentation.

The content of Web applications is modelled in a conceptual model where the classes of the objects that will be used in the Web application are represented by instances of <<conceptual class>> which is a subclass of the UML Class. Relationships between contents are modelled by UML associations between conceptual classes. During the navigational design, the navigation model is built. The navigational model is based on the conceptual model and represents the navigation paths of the Web application being modelled. A navigation model can be enriched by the results of the process modelling which deals with the business process logic of a Web application and takes place in the process model. The presentation model is used to sketch the layout of the Web pages associated to the navigation nodes.

### ***Adaptivity in UWE***

UWE provides a reference model for adaptive hypermedia systems (Munich reference model), based on the Dexter reference model [Halasz and Schwartz, 1990]. The purpose of this model is to identify the main features of adaptive hypermedia and personalized Web applications, as a prior step to the definition of appropriate modelling techniques. It is an object-oriented reference model, visually represented by UML and formally specified in OCL. It supports adaptation, adaptivity and even proactivity features. This reference model, similarly as the Dexter Model, is divided into three layers (within-component layer, storage layer, and run-time layer), where the storage layer contains the user model, the domain model, and the adaptation model as we can see in figure 2-5.

UWE stresses on personalization features, like the definition of a user model and an adaptation model, or a set of adaptive navigation features, which depend on preferences, knowledge or tasks that the user must execute.

- The static user model is represented as a class diagram, expressing the view the system has of the user. It includes the user attributes (and their values) relevant to the adaptive application. These attributes can be domain dependent (will have a value for each domain component) or domain independent attributes. State transitions of the objects (i.e. instantiation of the user model) can be expressed by rules of the adaptation model.

- The adaptation model consists of containers for user behaviour, a set of rules and a set of functions to perform the adaptation functionality. A rule is modeled as a class *Rule* that consists of one *Condition* one *Action* and *attributes*. The *executor()* method of the rule class performs the actions of rules. Rules are triggered by user behaviour (i.e. browsing, user input or user inactivity) or by other rules and they are based on the information of the user and domain models (as well as the user interaction activities). Depending on whether the rule is applicable to all instances of a domain class or just to a specific instance, two types of rules are distinguished: global or generic rules and local or specific rules. Rules are also classified depending on their objectives into *construction rules*, *acquisition rules* and *adaptation rules*. The construction rules find the appropriate concept on the basis of relationships. The acquisition rules objective is to fill the user model gathering the (user) needed information. The adaptation rules specify conditions under which to adapt the content, navigation and presentation of a Web application. UWE distinguishes three levels of adaptation: content adaptation, link or navigation adaptation and presentation adaptation.

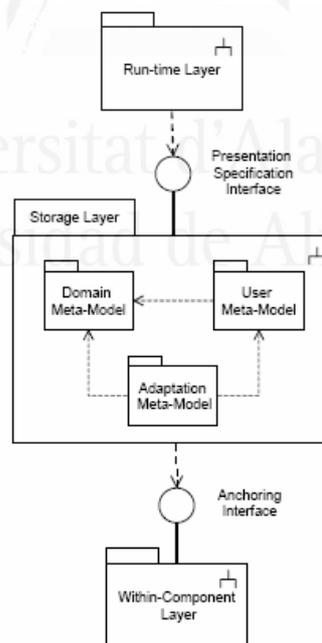


Figure 2-5: Architecture of Adaptive Hypermedia Applications

- The *adaptive content* selects different information depending on the current state of the user model. These rules can also update the information in the user model (i.e. acquisition rules).
- The *adaptive navigation* consists of changing the navigation structure of the Web applications, like changing the link appearance, the link target or the number of links presented to users, as well as sorting them. In the navigation model, adaptive navigation can be performed by direct guidance, sorted links, removed links, annotated links and passive navigation (specified by means of adaptation rules). For this purpose, associations showing navigability and access primitives (i.e. menus, guided tours, indexes) are annotated with properties to specify their adaptive behaviour (depending on the relevance they have for the user). In the navigation structure model (and also in the presentation model), these properties are annotated (just to specify their adaptive behaviour) based on the current state of the user model.
- The *adaptive presentation* shows different layouts or different font size, image size... etc depending on the conditions specified in the adaptation rules.

The introduction of the aspect-oriented modelling paradigm [Gray, 2003] and its consecutive acceptance by the research community leads to a number of extensions of existing design methods including UWE with this dimension. This paradigm supports separated modeling of different aspects that can be applied to an original model. UWE has been extended to support the use of aspects for modelling adaptive Web applications separating the navigation model from the adaptation model. For this purpose the UWE meta-model is extended with aspect meta-classes divided into runtime aspects and model aspects, first for modeling aspects that do change in runtime, second for those which do not change in runtime. UWE considers the link aspect introducing navigation annotations to particular links for link reordering, annotating or hiding, due to user behaviour. Content and presentation adaptation are also planned to be described by aspect-oriented modelling techniques.

### 2.3.5 Hera

The Hera methodology [Houben et al, 2004] is a model-driven methodology for designing and developing Web Information Systems (WISs). From the

gathering of requirements to the maintenance of the operational application, most information system design methodologies distinguish several phases in the design process. The development of a WIS is different in several aspects, and these aspects are the central focus of the Hera project. Like other WIS methodologies, Hera includes a phase in which the hypermedia (Web) navigation is specified. Hera considers navigation in connection to the data-intensive nature of the modern WIS in its *presentation generation* phase. Before, Hera's *integration and data retrieval* phase considers how to select and obtain the data from the storage part of the application. This includes transforming the data from these sources into the format (syntax and semantics) used in the application. Also, the handling of the interaction from users is specified: querying, navigation, or application-specific user interaction.

Hera's aim is to facilitate the automatic execution of the design: it should be possible to program the WIS in such a way that it can automatically execute the process specified by the design. Hera uses several models to capture the different design aspects. Because these models are considered Web metadata descriptions that specify different aspects of a WIS, they chose to use the Web metadata language, i.e. RDF(S) [Resource Description Framework], to represent all models and their instances. The choice is also justified by the RDF(S) extensibility and flexibility properties that enabled allow us to extend the language with model specific primitives to achieve the desired power of expression. As RDF(S) doesn't impose a strict data typing mechanism it proved to be very useful in dealing with semistructured (Web) data. Figure 2-6 shows how Hera typically views a WIS architecture:

- The *Semantic Layer* specifies the data content of the WIS in terms of a Conceptual Model (CM) which defines the concepts specific to the application domain. It also defines the integration process that gathers the data from different sources by means of the Integration Model (IM).
- The *Application Layer* specifies the abstract hypermedia view on the data in terms of an Application Model (AM) representing the navigation structure provided in the hypermedia presentation. Basic element in this model is a slice, a meaningful presentation unit that groups attributes from possibly different CM concepts. Slices can be recursively defined.
- The *Presentation Layer* specifies the presentation details that (with the definitions from the Application Layer) are needed for producing a

presentation for a concrete platform, like HTML, WML, or SMIL. The Presentation Model (PM) basic element is the region, an abstraction of a rectangular area of the user's browsing device to be used for presenting the information contained in a certain slice. Regions have associated with them layouts (e.g., a table layout or a vertical/horizontal layout) and a certain style (e.g., font size, font colour, link colour etc.). Regions can be recursively defined.

Providing clear relationships between the different models, e.g. by expressing one model in terms of another one, gives a major advantage: model-driven transformations. Populating the different models with data and then transforming them according to the relationships between the models leads to an automatic execution of the design at instance level, and ultimately to the production of the hypermedia presentation. The models in the phase of integration and data retrieval obtain the data from the sources. In reaction to a *user query* a *conceptual model instance* is produced with the data for which the application is going to generate a presentation. The models in the phase of presentation generation generate a hypermedia presentation for the retrieved data. The result of the user query, represented by the *conceptual model instance*, is transformed into a presentation in the specific format of the user's browser (e.g. HTML, WML, or SMIL).

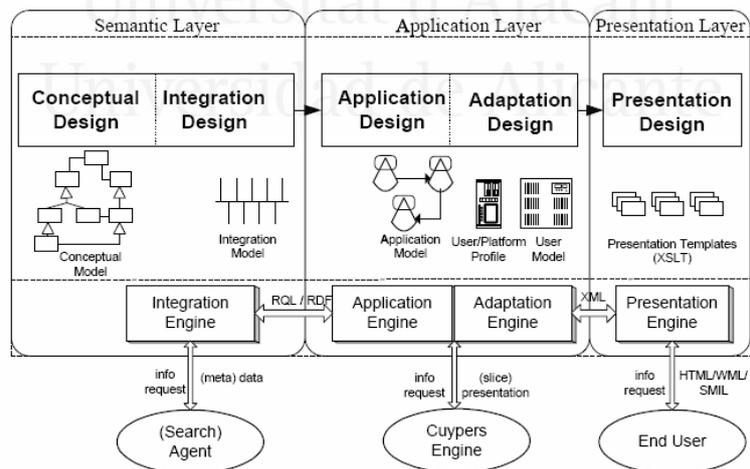


Figure 2-6: Hera Overview

### *Adaptivity in Hera*

There is also an orthogonal layer, the *Adaptation Layer* that captures adaptation issues in all the above layers [Frasincar et al, 2004]. Hera can filter certain concepts or attributes from the *conceptual model*. Adaptation can also be at the application model, suppressing certain slices and links that point to them. Moreover at the presentation layer, the presentation generation takes into account preferences and the device capabilities (i.e. platform being used, e.g. PC, PDA...) [Fia et al, 2004]. Different styles can also be chosen. At runtime presentation can dynamically change (using the same mechanisms as above). Also the user can be presented with information depending on his browsing/interaction history (e.g. forms, link following) [Houben et al, 2004; Vdovjak et al, 2003].

Hera supports two types of adaptation:

- **Static** based on the user profile data, that is stored in the *User/platform profile model (UP)*, which stores the *static* (i.e. fixed at design time) user preferences and platform capabilities.
- **Dynamic** based on the *User/platform model (UM)* information. In this model the dynamic user preferences and platform capabilities are captured. The *User model* changes during user browsing.

The data acquisition for the construction of the UM and the UP are outside the scope of Hera. In this layer there is another model called *adaptation model*, to specify how adaptation is done using the data coming from the UP or UM. This is done using by inclusion conditions attached to model elements. Conditions based on the UP attributes define the so called *adaptability* (presentation is fixed before user browsing) while the conditions based on the UM model define the so called *adaptivity* (presentation changes during user browsing) of the system [Frasincar and Houben, 2002; Vdovjak et al, 2003].

Information specified in the adaptation layers is taken into account when generating the actual implementation for a WIS. The Hera methodology proposes a sequence of data transformations that leads to a Web presentation. Based on the static part of the adaptation model the four basic models are tailored to the particular user/platform situation before the presentation is generated. Next, there is a sequence of pipeline transformations (the output of one transformation is the input of the subsequent transformation) to generate different model instances.

### 2.3.6 Other Approaches

There are other many Web design methods supporting (in some way) adaptation. We briefly summarize how some of them (Araneus, W2000 and WUML) support adaptation.

Araneus [Atzeni et al, 1998; Mecca et al, 1999] is an approach oriented to the design and development of Web applications. It includes a development process centred in the authoring of the application and a proprietary notation. In this approach personalization is treated by means of user roles: they consider that there should be a personal page for every user, the users are organized into user groups. Each group will be defined as an actor in the use case diagram and will have associated a different navigational model.

W2000 [Baresi et al, 2001] is a framework for the design of Web applications by means of combining UML elements with HDM elements [Garzotto et al, 1991; Garzotto and Paolini, 1993]. In this approach it is considered that different users have a different view of the application in terms of contents and operations. W2000 organizes the design activity in a set of independent tasks. Each activity produces a model that describes some aspects of the Web application. In the activity of *visibility design* it is specified to whom should be visible certain operations, information structures and navigational paths. Therefore this approach supports personalization only in base of user roles. Different application views are assigned to different user roles, in which is specified what is available for each of them.

WUML [Kappel, 2001], in the context of ubiquitous computing, has personalization based on context. The authors propose an object oriented framework. The framework consists out of four models: the context, profile, rule and event model. The context and profile model can be extended by the designer, and provide detailed information about the environment of an application and trigger the actual customization as soon as the environment changes. Context represents current and historical information about the environment of the application which is automatically monitored. Profiles cover more stable information which is explicitly given by a designer or a user (e.g. user preferences) or is transparently acquired by the system itself (e.g. usage statistics). Customization rules, in terms of the

event/condition/action (ECA) mechanism, are used for specifying the actual customization. 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.

## 2.4 Concluding Remarks

In this chapter we have studied the main concepts related to the hypermedia systems. Traditional hypertext and hypermedia systems do not take the individual users' features into account. They suffer from inherent navigation and orientation problems. Adaptive Hypermedia Systems can make hypertext more useful and usable personalizing the information according to the users' needs and preferences. AHS are being used in many application areas, but most of them target educational applications.

The World Wide Web evolution from a small set of (static) linked pages towards complex Web applications has lead to user disorientation and comprehension problems, as well as development and maintenance problems for designers. In most cases, development of Web-based systems has been ad hoc, lacking systematic approach and quality control and assurance procedures. Web Engineering, an emerging new discipline, advocates a process and a systematic approach to development of high quality Web-based systems. In this context Web Design Methodologies appeared, giving solutions both for designers and for users. However, new challenges appeared, like the need of continuous evolution, or the different needs and goals of the users. Adapting the structure and the information content and services for concrete users (or for different user groups) tackles the aforementioned (navigation, comprehension and usability) problems. It has therefore become a vital feature of modern Web applications.

Two (main) forms of adaptation exist: adaptability (i.e. adaptation initiated by the user, also called static adaptation) and adaptivity (i.e. adaptation initiated

by the system, also called dynamic adaptation). Adaptivity can be done for all the users or for a specific user (in the latter case it is called personalization). Web Site Design Methodologies in the context of adaptation focus mainly on personalization, either by adapting the content, the structure of the Website or the presentation. The exception is the *WSDM* approach, which doesn't consider personalization (i.e. adaptivity for a single user), but adaptivity. *Araneus* and *W2000* do not consider adaptivity features, they only support static adaptation (i.e. adaptability).

**Table 2-1:** Adaatation support in different methodologies

<b>Method</b>	<b>Adaptability</b>	<b>Adaptivity</b>	<b>Personalization</b>
<b>Araneus</b>	✓	✗	✗
<b>W2000</b>	✓	✗	✗
<b>WSDM</b>	✓	✓	✗
<b>OOHDM</b>	✓	✓	✓
<b>WebML</b>	✓	✓	✓
<b>UWE</b>	✓	✓	✓
<b>Hera</b>	✓	✓	✓
<b>WUML</b>	✓	✓	✓
<b>A-OOH</b>	✓	✓	✓

Regarding the Web Site Design methods supporting adaptivity they use different information to base the adaptivity on. Some of them focus on user characteristics, other focus on the environment information, others on the usage data... They way of modeling the actual adaptation also varies among the different methodologies, some use conditions to specify design alternatives, other use rules to specify the adaptive behaviour, other methods use queries over the data model, etc.

The approach presented in this dissertation (A-OOH) supports both adaptability, adaptivity and personalization. The aim of this approach is to extend the OO-H approach with these features (i.e. adaptivity and personalization). For this purpose an abstract personalization language has been defined called PRML (*Personalization Rules Modeling Language*). For this purpose, two conformance levels are defined in this language:

- *PRML Lite*: This level comprises the basic operations supported by the most representative Web modelling methods [Casteleyn et al, 2003; Ceri et al, 2003; Gómez et al, 2001; Houben et al, 2004; Koch, 2001; Rossi et al, 2001]. The purpose of its definition is being able to specify a (universal) reusable personalization policy.
- *PRML Full*: With the constructs defined in this level more complex personalization strategies can be specified. PRML Lite is a subset of PRML Full; therefore the particularities defined in PRML Lite are also valid for PRML Full.

The personalization actions supported in this approach are the following:

- **Actions over attributes (User and Navigation Models):**
  - *Updating an attribute value from the User Model*: This action allows modifying/setting the value of an attribute (of a concept) of the User Model.
  - *Filtering attributes in the Navigation Model nodes*: By means of this action a node can be restricted by hiding or showing some of the attributes of the Domain Model/User Model related concept.
- **Actions over links (Navigation Model):**
  - *Hiding links and their target nodes*: Analogous to filtering data content, PRML also supports filtering links.
- **Actions over nodes (Navigation Model):**
  - *Filtering node instances*: This action only shows the selected instances of a Domain Model/User Model concept for a user depending on the personalization requirements we want to support.
  - *Sorting node instances*: In PRML, node instances can be sorted by a certain value to satisfy a personalization requirement.

The adaptation of the presentation is not yet considered in this approach.



Universitat d'Alacant  
Universidad de Alicante

## Chapter 3

### The Adaptive OO-H Method

The Adaptive OO-H method (A-OOH) is an extension of the OO-H (Object Oriented Hypermedia) approach [Gómez et al, 2000; Gómez et al, 2001] to support the modelling of adaptive (and personalized) Web applications. It supports most of the OO-H basic features, but some updates and extensions have been added for the support of adaptive Web sites modelling. These updates and extensions done are explained along the chapter as well as in the last section of it. The same as OO-H, A-OOH is a user-driven methodology based on the object oriented paradigm and partially based on standards (XML, UML, OCL...). The approach provides the designer the semantics and notation needed for the development of adaptive Web-based interfaces and their connection with pre-existing application modules.

It is important to note that OO-H (and A-OOH) is not a methodology that supports the full development cycle of the application. Some aspects like the planning, risk management or processes integration for quality evaluation are not considered as development phases. However, the presented approach is focused on the authoring of the application: requirements, design, implementation and maintenance and evaluation.

The outline of the chapter is as follows. The design process proposed by A-OOH is presented in Section 3.1. In Section 3.2 the different models of the A-OOH approach are explained. Finally in Section 3.3 the conclusions of the chapter are presented and the main extensions done to the OO-H method are summarized.

### 3.1 Design Process

Adaptive hypermedia systems are complex systems which require an appropriate software engineering process for their development. A-OOH<sup>1</sup> design process is based on the Unified Software Development Process (USDP), also known as Unified Process or UP [Jacobson, 1999] and on a popular refinement of it, the Rational Unified Process [Kruchten, 2004]. The UP is a popular iterative and incremental software development process framework which specifies how to develop software using UML. The UP is not simply a process, but rather an extensible framework which can and should be customized (instantiated) for specific organizations and/or projects. The most important characteristics of the UP are described next:

- **Use-Case and Risk Driven** - The process employs Use Cases to drive the development process from inception to deployment. The Unified Process requires the project team to focus on addressing the most critical risks early in the project life cycle.
- **Architecture-Centric** - The process seeks to understand the most significant static and dynamic aspects in terms of software architecture. The architecture is a function of the needs of the users and is captured in the core Use Cases.
- **Iterative and Incremental** - The development of the software products are defined into a series of timebox iterations. Each iteration results in an increment, which is a release of the system that contains added or improved functionality compared with the previous release. An iteration may encompass all of the workflows in the process. The iterations are planned using Use Cases.

The Unified Process consists of cycles that may repeat over the long-term life of a system. A cycle consists of four phases: Inception, Elaboration, Construction and Transition. Each cycle is concluded with a release, there are also releases within a cycle. A-OOH design process considers an extra phase: Maintenance. This extension was considered in the UWE approach [Koch, 2001] due to the importance of the maintenance in the life cycle of a hypermedia application. Maintenance implies changes in the content, in the

---

<sup>1</sup> OO-H design process was based on the spiral model.

hypermedia structure and in the layout. The five phases in a cycle are the following:

- **Inception Phase** - During the inception phase the core idea is developed into a product vision. The inception phase establishes the product feasibility and delimits the project scope. The milestone for this phase is defining the life-cycle objectives.
- **Elaboration Phase** - During the elaboration phase the majority of the Use Cases are specified in detail and the system architecture is designed. This phase focuses on the "Do-Ability" of the project. We identify significant risks and prepare a schedule, staff and cost profile for the entire project. The milestone for this phase is defining the life-cycle architecture.
- **Construction Phase** – This phase focuses on the development of the system. A fully operative and efficient product is elaborated. This phase is finished when all the use cases are implemented. The milestone of this phase is the initial operational capability.
- **Transition Phase** - In the transition phase the goal is to ensure that the requirements have been met to the satisfaction of the stakeholders. This phase is often initiated with a beta release of the application. Other activities include site preparation, and defect identification and correction. The milestone of the transition phase is the product release.
- **Maintenance Phase** – This phase starts when the system is built and delivered to the final users, and lasts during the life of the system. An adaptive Website needs continuously refresh and update, the designer can add new adaptivity requirements, update the content of the Website, adapt the Website to new technologies, etc.

The Unified Process identifies five core workflows that occur during the software development process (Requirements, Analysis, Design, Implementation and Test). In the RUP these core workflows are 6: Business Modeling, Requirements, Analysis and Design, Implementation, Test and Deployment. The workflows are not sequential and likely will be worked on during all of the phases. Each workflow is defined by a set of activities that are performed by a set of workers with the goal to produce some artifacts (it can be a model, a model element or a document), which are measurable results of the workflow. The workflows are described separately in the

process for clarity but they do in fact run concurrently, interacting and using each other's artifacts. A-OOH case considers the following workflows:

1. **Requirements:** In this stage the requirements for each type of user are gathered, including the personalization (adaptation) requirements.
2. **Analysis and Design:** In this stage all the activities related to the analysis and design of the software product are included:
  - a. *Domain Analysis:* From the user requirements and the designer knowledge of the domain, the relevant concepts for the application are gathered.
  - b. *Domain Design:* The domain analysis model has to be refined in consecutive iterations with new helper classes, attribute types, parameters in the methods... etc.
  - c. *Navigation Design:* The domain information is the main input for the design navigation activity, where the navigational paths are defined to fulfil the different functional requirements and the organization of that information in abstract pages<sup>2</sup>.
  - d. *Presentation Design:* Once the logic structure of the interface is defined, OO-H allows to specify the location, appearance and additional graphical components for showing the information and navigation of each of the abstract pages.
  - e. *Adaptation Design:* In parallel to the other sub-phases an adaptation design phase is performed, which allows to specify the adaptation (or personalization) strategies to be performed.
3. **Implementation:** Implementation is the following workflow considered in A-OOH where the final application is generated.
4. **Test:** The goal of this workflow is verifying that the implementation work as intended.

As a result of performing the activities of each of the design process phases, in A-OOH we get a set of models, reflecting views of the interface to be generated. A model can evolve along the different phases over time. In the next sections we describe these workflows in more detail, as well as the diagrams used in each workflow.

---

<sup>2</sup> An abstract page is a logic unit grouping content and links, and does not necessarily correspond with a physic (final) page.

## 3.2 Requirements

The development process in A-OOH starts with a workflow specifying the functional requirements (including the personalization requirements) similar to most we can find in methodologies for the design of other types of software applications. This activity of requirement gathering for every user type (i.e. user role) manages the rest of the activities of the development process. The functional requirements considered can be requirements related to the content, to the structure or to the presentation. The adaptation (personalization) requirements can also relate to the content, structure or presentation of the Website. Non-functional requirements (i.e. system properties, performance... etc) are not taken into account.

### 3.2.1 Use Case Diagram

OO-H proposes the Use Case Diagram (UCD) with UML notation and semantics for the gathering of functional requirements in the system. UCDs have only 4 major elements: The **actors** that the system you are describing interacts with, the **system** itself, the **use cases**, or services, that the system knows how to perform, and the lines that represent **relationships** between these elements. The associations between actors and use cases indicate that there is a communication between an instance of a use case and the user with the role indicated by the actor. Use cases can be related to each other by means of relationships of the type inclusion (<<include>>), extension (<<extend>>) and generalization. The steps that OO-H follows for constructing the UCD are:

- Identify the actors.
- For each actor, identify the activities performed in the system.
- Group the activities in use cases.
- Establish the associations between use cases and actors.
- Establish the generalizations needed to simplify the diagram among actors.
- Establish the relationships of the type <<include>> and <<extend>> among use cases.

There are several degrees of detail for writing use cases. Moreover the use case diagram can be enriched with several mechanisms (e.g. defining activity diagrams, etc). However in OO-H it is not needed to specify more than the association between actors and activities expressed in the use case diagram.

### 3.2.1.1 Example

We consider now an example of a (simplified) online video-club system. In this system there are three different types of users:

- *Anonymous user*: who will be able to register and consult movies.
- *Client*: who will be able to rent movies, change his personal details and consult personalized recommendations.
- *Administrator*: who will be able to manage the movies (add and delete) and manage the offers (insert, delete or modify).

Moreover, both the client and the administrator users will also be able to perform the functionality assigned to the anonymous user. The first task to build the use case diagram would be to identify the actors, from the system description we can see there are two actors (i.e. client and administrator) that inherit from the anonymous actor. For every actor we assign the previously identified activities that he is able to perform in the Website. In Figure 3-1 we can see the use case diagram of the system.

## 3.3 Analysis and Design

In A-OOH we consider one single workflow for analysis and design the same as in RUP. Design is considered as a refinement process of the analysis during the different iterations.

### 3.3.1 Domain Analysis and Design

As the result of the domain analysis and domain design phases the domain model (DM) is defined. It specifies the structure of the Web application domain data. It is expressed in OO-H as an UML compliant class diagram. It encapsulates the structure and functionality required of the relevant concepts of the application and reflects the static part of the system. In this moment the issues related to the navigation, presentation, workflow or interaction are not

taken into account yet, which simplifies and gives a bigger reuse capability to the model.

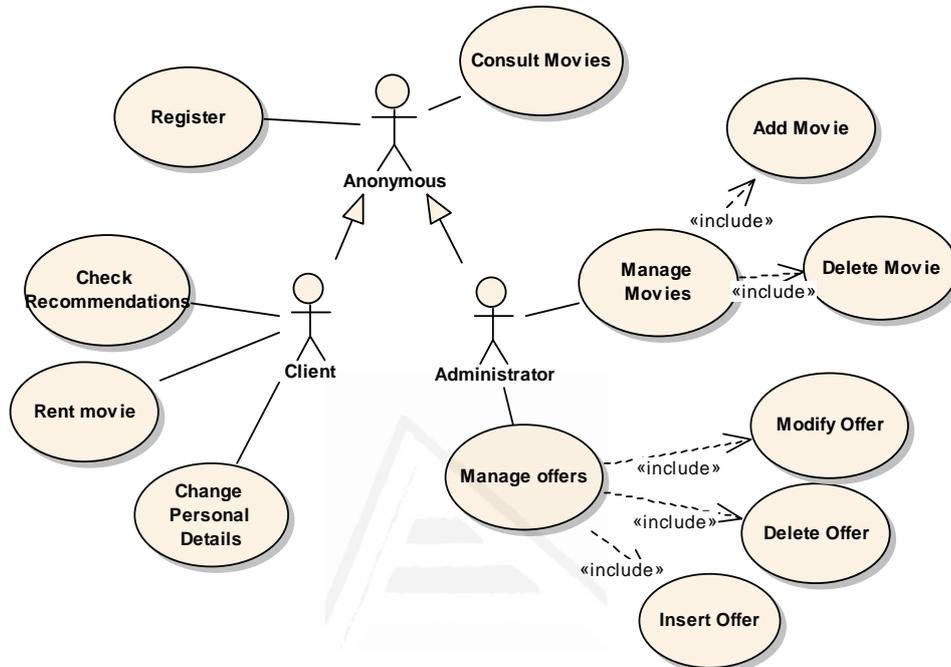


Figure 3-1: UCD example

The main modeling elements of a class diagram are the classes (with their attributes and operations) and their relationships (association, aggregation, composition and generalization). Its construction follows well-known object oriented modeling techniques:

- Identify the classes.
- Determine the association relationships among them.
- Define hierarchical relationships.
- Specify the most relevant attributes and operations.
- Identify the cardinalities.
- Include as classes the actors identified in the requirements workflow, if it is needed to store some kind of information.

### 3.3.1.1 Example

We consider now an example of an academic system with students and lecturers. Students can register for different courses taught by the lecturers, there is only one lecturer responsible for the course (if any). The courses have a set of requirements for registration. We store information about the students and about the lecturers. The Web application domain data is specified by means of the domain model, shown in Figure 3-2. Concepts are represented by classes with attributes. There are associations between them representing concepts relationships.

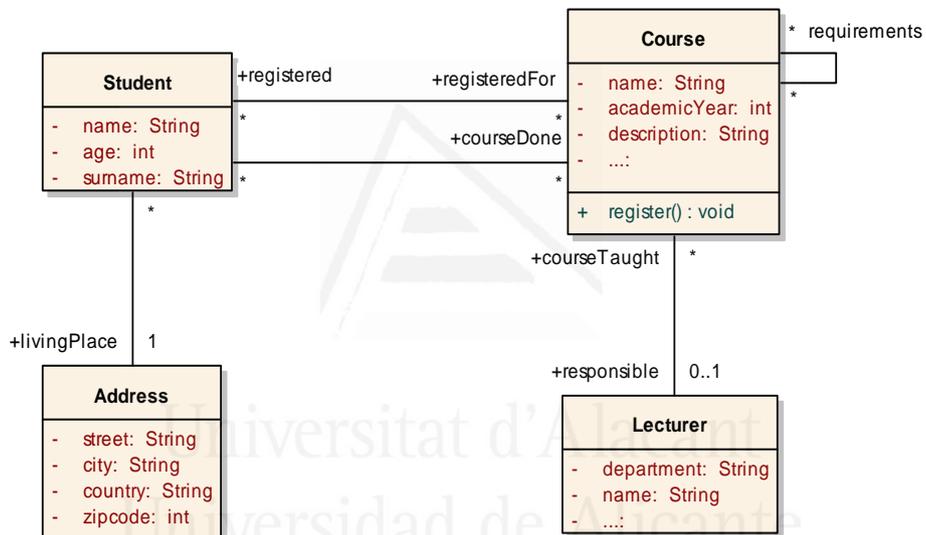


Figure 3-2: DM example

### 3.3.2 Navigation Design

Once the Domain Model (DM) has been specified the navigation structure is defined by means of the Navigation Access Diagram (i.e. NAD). This diagram enriches the DM with navigation and interaction features.

In A-OOH instead of using the notation of the NAD presented in OO-H, an extension of UML by means of a UML profile is used. This profile is defined by a set of stereotypes and tagged values to represent the NAD concepts in

UML notation. The main advantage of using a standard language (UML) is that the learning curve for the designer is smaller. In this section first we present the elements of the Navigational Model to understand the presented approach and then we present the UML MOF Metamodel and UML profile defined. Finally we show an example of the NAD.

### 3.3.2.1 Navigation Access Diagram

A Navigational Model (NM) describes a navigation view on data specified by the DM. In OO-H the NM is captured by one or more Navigation Access Diagrams (i.e. NADs). The designer should construct as many NADs as different views of the system are needed, and provide at least one different NAD for each identified (static) user role.

The NAD is composed of *Navigational Nodes*, which represent (restricted) views of the domain concepts, and their relationships indicating the navigation paths the user can follow in the final Website (*Navigational Links*). Each Node has associated a (owner) Root Concept from the DM attached to it by the notation: “Node:DM.RootConcept”. There are three different types of navigational Nodes:

- **Navigational Classes (NC):** These are domain classes enriched with attributes and methods which visibility has been constrained depending on the access permissions of the user and the navigational requirements. An example of this enrichment is the differentiation among three attribute types: V-attributes (visible attributes), R-attributes (referenced attributes, which are shown after a user request) and H-attributes (hidden attributes, only shown when an exhaustive view of the system population is required, e.g. for refinement of code reasons). It is represented by a UML class with the stereotype `<<NavigationalClass>>`.

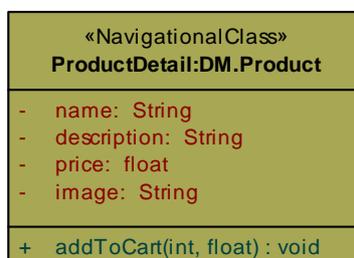


Figure 3-3: Navigational Class

- **Navigational Targets (NT):** They group the model elements which collaborate in the fulfilment of every navigation requirement of the user. NTs are represented by UML packages with the stereotype <<NavigationalTarget>>.

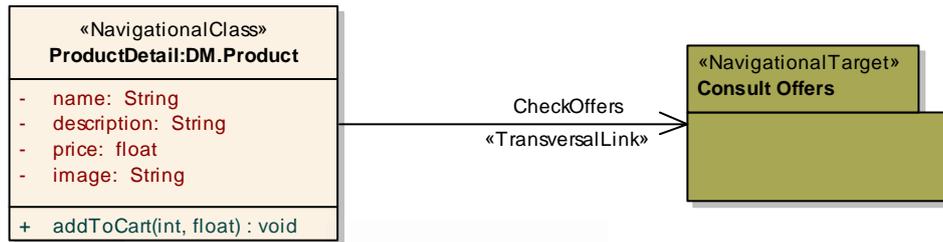


Figure 3-4: Navigational Target

- **Collections:** They are (possible) hierarchical structures defined in Navigational Classes or Navigational Targets. They provide the user new ways of accessing the information. The most common collection type is the C-collection (Classifier collection) that acts as an abstraction mechanism for the concept of menu grouping Navigational Links. Another important collection is the S-collection (Selector collection) with which we can represent a selection mechanism (for example the concept of form). It is represented by a UML class with the stereotypes <<NavigationalC-Collection>> or <<NavigationalS-Collection>>.

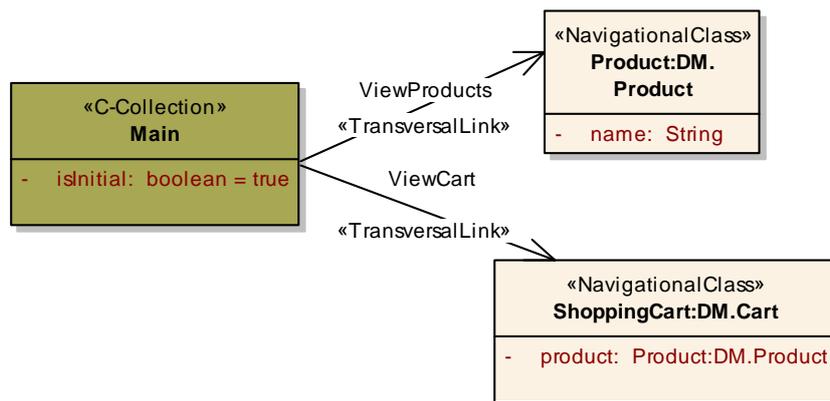


Figure 3-5: C-Collection

Every node has a boolean tagged value “IsInitial”, if its value is set to *true* indicates that the node is the entry point of the NAD or of a NT. By default its value is set to *false*.

Navigational Links (NL) define the navigational paths that the user can follow through the system. A-OOH defines two main types of links:

- **T-Links (Transversal Links):** They are defined between two navigational nodes (navigational classes, collections or navigational targets). The navigation performed is done to show information through the user interface, without modifying the business logic. This type of links is represented by the stereotype `<<TransversalLink>>`.

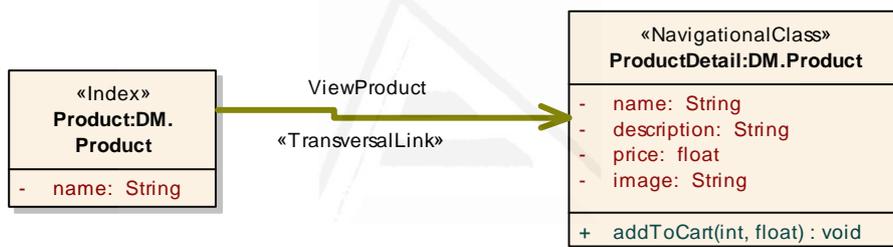


Figure 3-6: Transversal Link

- **S-Links (Service Links):** Navigation is performed to activate an operation which modifies the business logic and moreover implies the navigation to a node showing information when the execution of the service is finished. It is established when a service of the navigational class is activated. This type of links is represented by the stereotype `<<ServiceLink>>` and has associated the name of the invoked service.

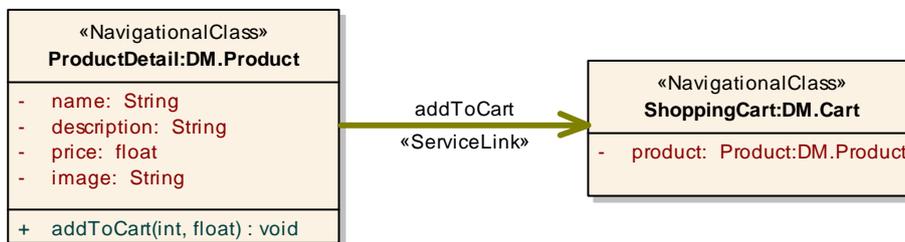


Figure 3-7: Service Link

- **Design Patterns:** When defining the navigation structure, the designer should take into account some orthogonal aspects such the desired behaviour of the navigation, the selection of the object population, the order in which objects should be navigated, or the access cardinality. These features are gathered by means of different types of navigation patterns and associated filters to links and collections. Links can have associated one of the three navigational patterns: *Guided Tour*, *Showall* and *Index*. Navigational patterns are characterized by two properties: indexed (yes/no and, if the value is true, number of elements in each page, to allow the pagination with indexes) and navigation (yes/no, and in case of “yes”, number of elements by page, to decrease the size of guided tours). Next we briefly describe the design navigation patterns supported by A-OOH:
  - *Index pattern:* it causes an indexed navigation of a set of objects which are presented in an index page. In this pattern internal navigation amongst the target objects doesn't exist, the only way to access the objects of the collection is by means of the index.
  - *Guided tour pattern:* it causes a non indexed navigation in which internal navigation exists and the objects of the navigation are presented one by one.
  - *ShowAll pattern:* it implies navigation without indexing and without internal navigation, all the objects are shown in the same abstract page. This is the default pattern in a Navigation Class.

In the NAD they are represented with an stereotype with the name of the design pattern (i.e <<Index>>,<<GuidedTour>>) excepting for the Showall pattern which is set by default in a Navigation Class. An example of the representation of the Index pattern is shown in the Figure 3-8.

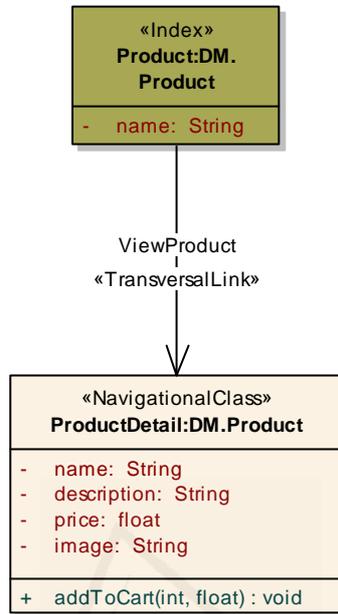


Figure 3-8: Index Design Pattern

The stereotypes for representing the different concepts described in this section are defined in the UML profile presented in Section 3.3.2.3. This UML profile extends the concepts defined in the NAD metamodel presented next.

### 3.3.2.2 NAD Metamodel

In Figure 3-9 the MOF metamodel defined for the NAD is shown. The main elements of the NAD metamodel are the Navigational Node and the Navigational Link.

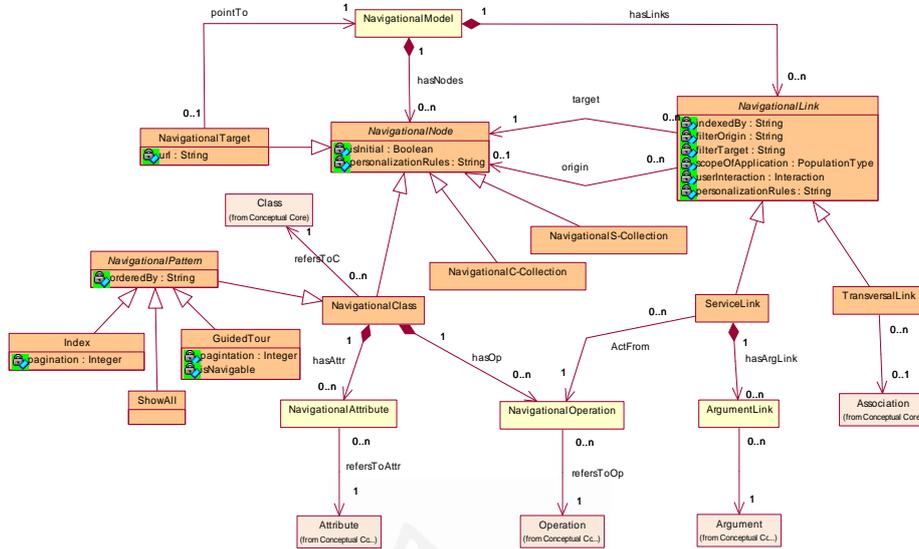


Figure 3-9: NAD Metamodel

In the metaclass representing the *Navigational Link* we can see that several attributes are defined (see Figure 3-10):

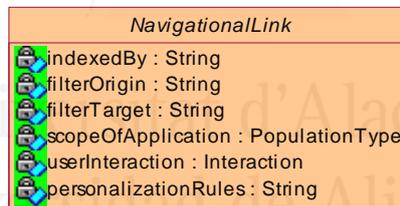


Figure 3-10: Navigational Link metaclass

- *indexedBy*  
This attribute indicates, in the case of an indexed navigation, the attribute by which the node instances are indexed.
- *filterOrigin, filterTarget*

To define navigation constraints, OO-H uses the object constraint language [Warmer & Kleppe, 1998], a subset of the standard UML that allows software developers to write constraints over object models augmenting the model precision. OO-H associates such constraints to the navigation model by means

of filters defined upon links. Filters can be defined over the origin or the target Navigational Class restricting the set of objects. In the NAD they are represented in the target or source role value (depending on the filter type), as it is shown in the Figure 3-11. The link *ProductDetail* must be active only for the products with a price greater than 50.

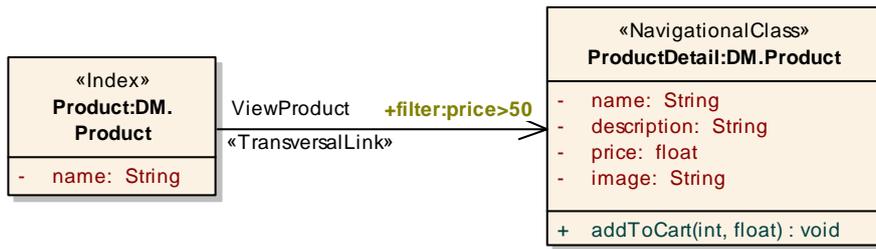


Figure 3-11: Filter defined in a Navigational Link

o *User interaction* (manual/ automatic)

Sometimes is useful for the user, not being obliged to click on a link to get the set of information (i.e. the navigation is activated by the system). This feature is captured in the attribute of user interaction in the metamodel, that in this case will take the value *automatic* instead of the *manual* traditional value (by default). The *automatic* property is represented by a composition as it is shown in the Figure 3-12, meanwhile the *manual* property is represented by means of an association

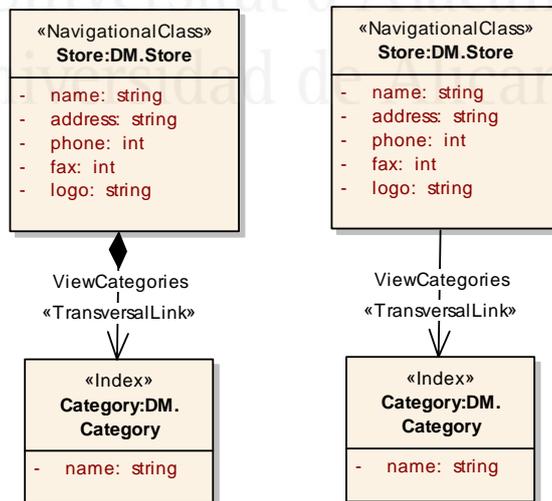


Figure 3-12: Automatic and Manual Links

- *scopeOfApplication*

This attribute is also defined in the *NavigationalLink* metaclass. The Navigational Links can have additional information to specify the application field /scope of application (simple / multiple / universal). In the case of the links that transfer information about the origin of the navigation three different cases can be given:

- **Simple**: the object from which the navigation departs is a single object. This is the default value in the Transversal Links.
- **Multiple**: the origin page shows a set of objects from which the user selects a subset at execution time.
- **Universal**: the link has to transfer all the information referent to the whole set of objects in the origin page. This is the default value in the Service Links associated with a method of a class.

The attribute *scopeOfApplication* can then have one of these three values (i.e. *simple*, *multiple*, *universal*).

In the case of the *NavigationalNode* metaclass, we can see the four types of Navigational Nodes defined in Figure 3-13. The *NavigationalTarget* metaclass has the attribute *url* which contains the Url of the target page in the case of being an external page (i.e. not defined in the system).

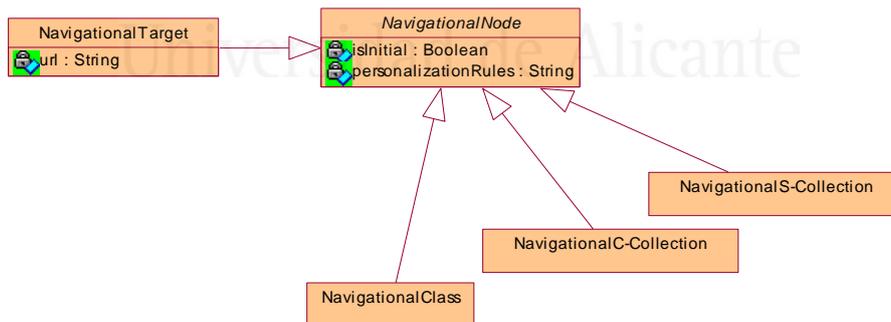


Figure 3-13: Navigational Node metaclass

Personalization can be added to the NAD by means of personalization rules. These rules can be attached either to *Navigational Nodes* or to *Navigational Links*. The rules attached are stored in the *PersonalizationRules* tagged value. It is explained in the *adaptation design workflow* in Section 3.3.4.

Next the UML Profile defined for the NAD is presented.

### 3.3.2.3 UML Profile for the Navigation Access Diagram

As aforementioned a profile for the NAD has been defined. Extending the UML concepts of *class* and *association* the NAD concepts are specified. The purpose of defining an UML profile is to provide an easy mechanism of adaptation to the UML metamodel to elements that are specific of a particular domain, platform or method. In this sense, the particular profile for the NAD consists in adapting the elements defined in the NAD to the UML metamodel. In this way the profile allows to specify the NAD elements in any commercial tool that supports UML.

- The NAD Navigational Node is defined as an extension of the *UML class concept* which has attributes and operations (also extensions of the UML concepts). There have been defined different stereotypes for representing the different types of Navigational Nodes (i.e. <<NavigationalClass>>, <<NavigationalC-Collection>>, <<NavigationalS-Collection>>) in the final model.
- Regarding the Navigational Class concept the following information is stored:
  - The Navigational Class has information about the name of the root concept in the Domain model (stored in the tagged value *rootConcept*).
  - Moreover if a Navigational Class has associated a design pattern different to the default one (i.e. Showall), the stereotype for representing the concept is the name of the design pattern (i.e. <<Index>> or <<GuidedTour>>).
- The NAD Navigational Target concept is defined as an extension of the *UML package concept* and the stereotype <<NavigationalTarget>> is defined to represent this concept.
- Moreover the tagged value *isInitial* is defined over the NavigationalNode concept and over the Navigational Target concept to indicate whether that is the initial point of the navigation.

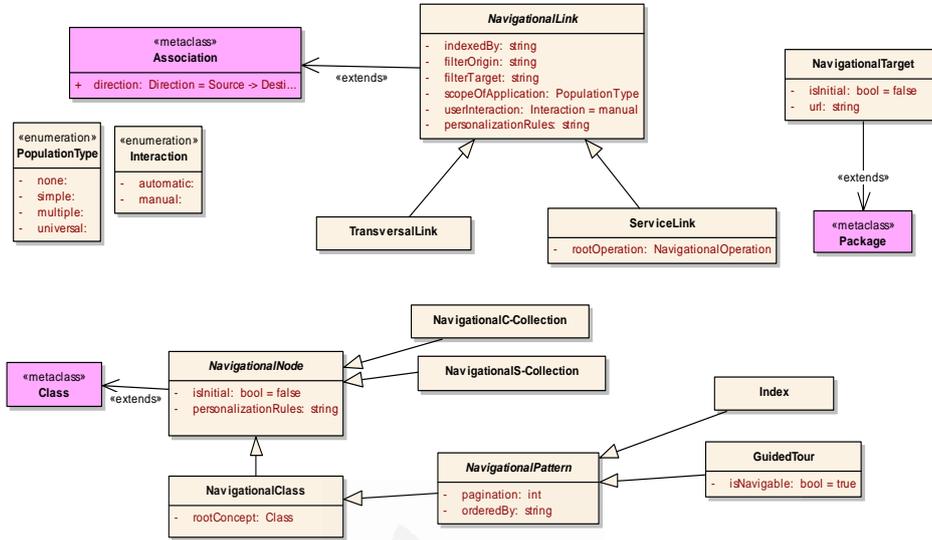


Figure 3-14: UML Profile for the NAD

In the next section an example of the NAD is shown.

### 3.3.2.4 Example

For illustrating the Navigation Access Diagram, we consider a (simplified) virtual shop system. Users can consult and buy products, check the products by category and consult the shopping cart.

In the NAD shown in Figure 3-15 we can see the initial page is a Navigational Class (which has the `isInitial` tagged value to true). This Navigational Class has an automatic link `ViewCategories` and a navigational link `ViewProducts`. The target of both links is a Navigational Class with the `Index` navigational design pattern, which causes an indexed navigation of the categories and products respectively. Both indexes are ordered by name (indicated in the corresponding tagged value). Product details can be consulted and they can be added to the shopping cart (see the ServiceLink `addToCart`). The products in the shopping cart can also be consulted. Moreover in this system we have a navigational target (`order`) which navigation is not specified here.

Once the NM has been defined during the Navigation Design, the Presentation Design is the next workflow to perform.

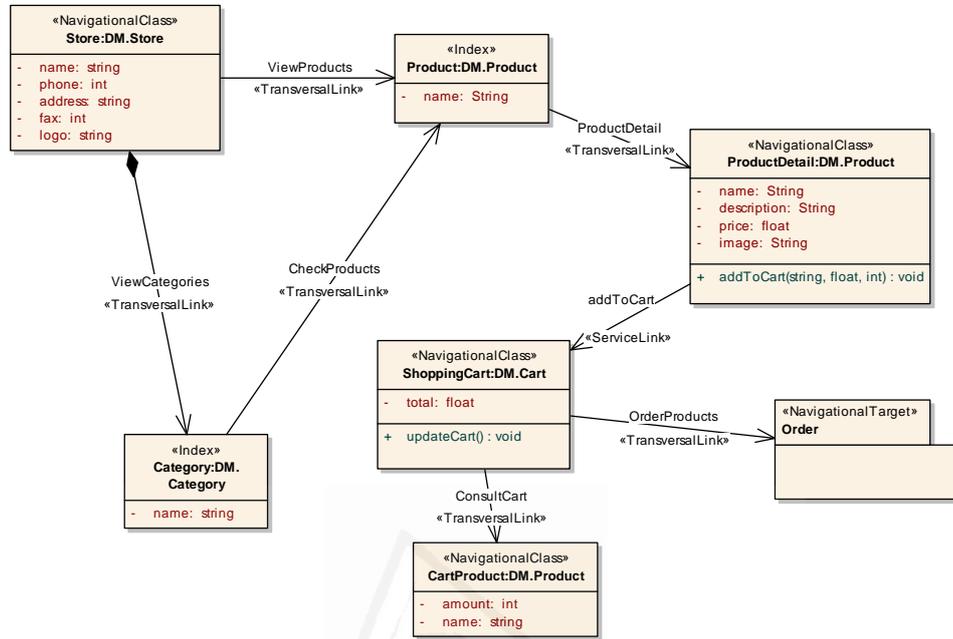


Figure 3-15: NAD example

### 3.3.3 Presentation Design

During the presentation design, the concepts related with the abstract structure of the site and the specific details of presentations are gathered.

The Presentation Model is defined in this activity. It is captured by one or more Design Presentation Diagrams (i.e. DPDs). There should be one DPD for each NAD defined in the system. This diagram enriches the Navigation Action Diagram described in previous section. The DPD uses UML notation. In this section the DPD is presented describing its main elements, then the MOF metamodel in which the DPD is based is described. Finally the UML profile defined to extend the UML concepts in order to represent the DPD elements is presented. An example is shown in section 3.3.3.4 for a better understanding of the DPD.

#### 3.3.3.1 Design Presentation Diagram

The DPD describes *how* the navigation elements are presented to the user. The DPD main objectives are:

- To provide the page structure of the Website, grouping the NAD Navigational Nodes into Presentation Pages. These Presentation Pages are *abstract pages*, which in the final implementation can be represented by one or more concrete pages. The designer can add static pages also directly on the DPD. This is represented in the level 0 of the DPD.
- The second goal is to describe the layout and style of each page of the interface. The designer should decide which interface components are going to be in the page and where are they going to be positioned. Moreover, he can modify the individual structure of the pages and add static elements directly on the DPD. This is represented in the level 1 of the DPD, exploding each of the *abstract pages* previously defined in the level 0.

Once the DPD has been refined, a Web application front-end, either static or dynamic, can be generated for the desired environment (HTML, WML, ASP's, PHP's...) depending on the restrictions of the target platform.

The main modelling elements of the DPD are described next. We classify them into level zero or level one depending on where they can be defined.

#### **Main elements defined in the level zero of the DPD:**

As aforementioned this level provides the page structure of the Website, grouping the NAD Navigational Nodes into (abstract) Presentation Pages. Moreover new static pages can be added to this level.

- **Presentation page**

This element represents an abstract page which has associated a Presentation model where are defined all the components shown in the page. It is represented as a UML Package with the stereotype <<PresentationPage>> (see Figure 3-16). Inside this package the Presentation Model attached to the Presentation Page is shown.

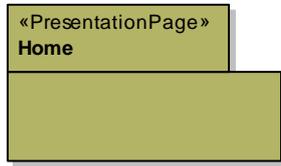


Figure 3-16: Presentation Page

- **Page Chunk**

The Page Chunk element represents a fragment of an abstract page which has associated a Presentation model where the components shown in this fragment are defined. This fragment can be reused in the different pages that compose the Web application. In this way we avoid to make several diagrams of common parts to all (or many of) the pages. It is represented as a UML Package with the stereotype <<PageChunk>> (see Figure 3-17). Inside this package the Presentation Model attached to the Page Chunk is shown.

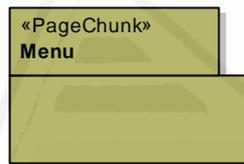


Figure 3-17: Page Chunk

- **Window**

This element represents the window of the Web browser that is being used. This element has been defined to give the chance to the modelled Web application of using several windows of the browser during the use of the application. The notation is a UML class with the stereotype <<Window>> as we can see in Figure 3-18.

- **FrameSet, Frame**

The FrameSet element represents a set of frames in which the browser window can be divided. In this way we provide the designer the possibility of using a frame based design for his application. It is represented as a UML class with the stereotype <<FrameSet>> as seen in Figure 3-18.

The Frame element represents a frame that is part of a FrameSet. It is represented with a UML class with the stereotype <<Frame>>. FrameSet and Frame elements are associated with the <<includeFrame>> association.

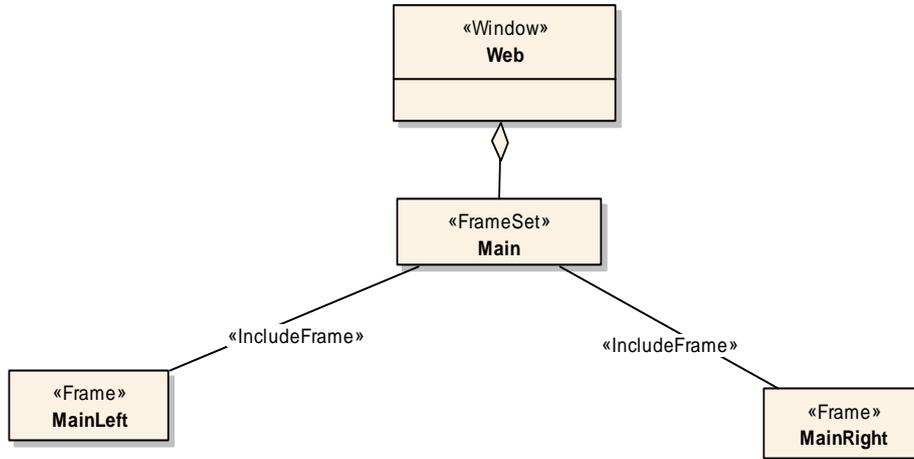


Figure 3-18: Window, FrameSet and Frame elements

### **Main elements defined in the level one of the DPD:**

As already explained, this level goal is to describe the layout and style of each page of the interface (defined in the level 0). Static elements can be added to the pages (e.g. static text).

- **Layout**

Instead of using frames, layouts can be defined for defining the disposition of the objects visualized in the Web page. These layouts can be of three different types: *BoxLayout*, *BorderLayout* and *GridBagLayout*; each of them provides a concrete distribution for its cells. The layouts are translated, in last instance, to HTML tables. The different types of layouts considered can be nested.

- *BoxLayout*

*BoxLayout* either stacks its components on top of each other or places them in a row — your choice.



Figure 3-19: BorderLayout

- *BorderLayout*

As the Figure 3-20 shows, a BorderLayout has five areas in which we can place the different elements of a Web page.

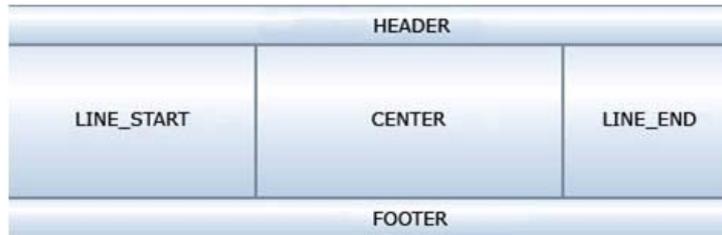


Figure 3-20: BorderLayout

- *GridBagLayout*

GridBagLayout is one of the most flexible layout. A GridBagLayout places components in a grid of rows and columns, allowing specified components to span multiple rows or columns. Not all rows necessarily have the same height. Similarly, not all columns necessarily have the same width. Essentially, GridBagLayout places components in rectangles (cells) in a grid, and then uses the components' preferred sizes to determine how big the cells should be. The following figure shows an example of a gridbaglayout. As you can see, the grid has three rows and three columns. The button in the second row spans all the columns; the button in the third row spans the two right columns.



Figure 3-21: GridBagLayout

- **Cell**

A cell represents a component which is part of a layout. The cells act as containers for the interface components.

- **InterfaceComponent**

An interface component presents the visualization of an object (text, images, links). The interface components considered are:

- Composed Interface Component

- Anchor

Anchor is a composed interface component which contains a simple interface component.

- Simple Interface Component

The simple interface components considered are:

- Image, Text , FormElement (e.g. InputButton, InputSubmit...etc).

The stereotypes used in the DPD notation are defined in the UML profile presented in Section 3.3.3.3. This UML profile extends the concepts defined in the DPD metamodel presented in the next section.

### 3.3.3.2 DPD Metamodel

The DPD MOF metamodel has been defined to formalize the elements of the DPD and the existing associations among them. A metamodel defines the language to express the model.

The main elements of a DPD are the Presentation Nodes and the relationships among them (i.e. Presentation Links).

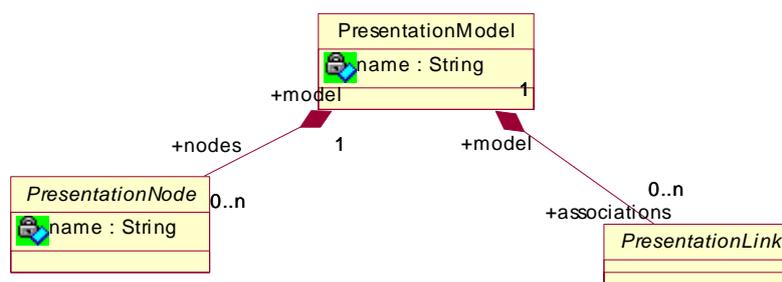


Figure 3-22: DPD Main Elements

Two types of presentation nodes are considered depending on the level of the DPD.

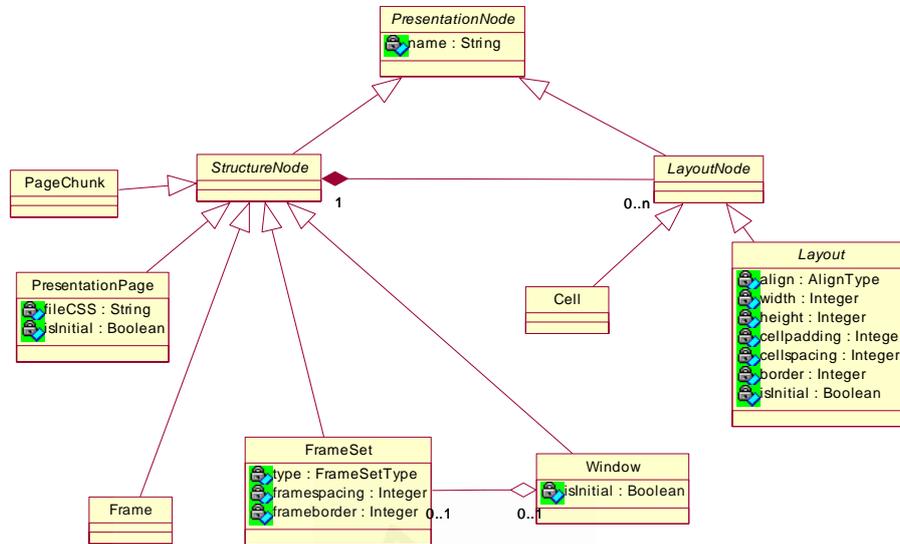


Figure 3-23: Presentation Node Subtypes

The different types of Structure Nodes considered are: Presentation Page, Window, Frame, Frameset and Page Chunk (level zero of the DPD). The types of Layout Nodes considered are: Layout and Cell (level 1 of the DPD).

In a Presentation Model five types of Presentation Links can be defined (see Figure 3-24):

- *Navigates*: This relationship can be defined between Presentation Page elements.
- *Contains*: This relationship is defined between Presentation Page elements and Page Chunk elements to indicate a presentation chunk is contained (and shown) in one or several Presentation pages.
- *Presents*: This relationship can be defined between Frame elements and Presentation Page elements.
- *IncludeFrame*: This relationship can be defined between FrameSet and Frame elements.
- *IncludeCell*; This relationship is defined between Layout and Cell elements.

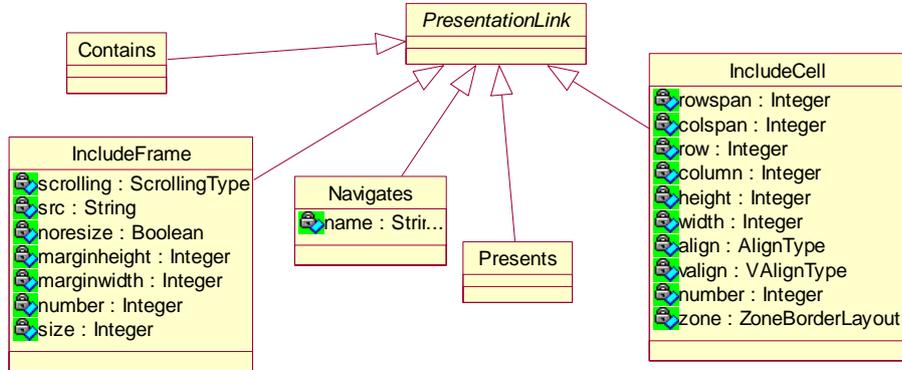


Figure 3-24: Presentation Link types

The Layout elements have associated information like the alignment, width, height...They can have associated one or more cells. In Figure 3-25 the metaclasses representing the different layout types are shown. In this figure we can also see that the association “IncludeCell” (defined between Layout and Cell elements) contains information about the positioning and size of the elements inside the cells.

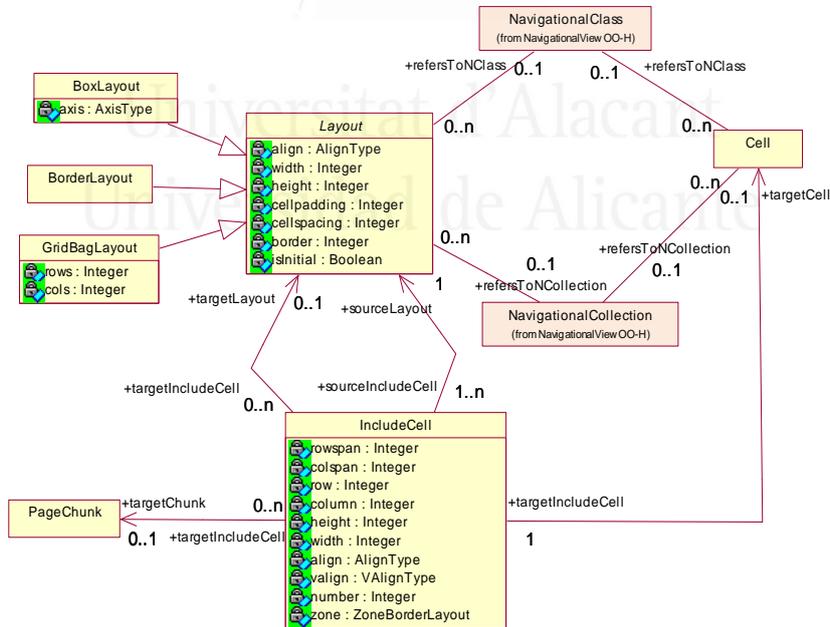


Figure 3-25: Layout and IncludeCell metaclasses

### 3.3.3.3 UML Profile for the DPD

Once the DPD MOF metamodel has been defined, a UML profile is presented from the DPD model for expressing it in UML 2.0.

- The DPD Layout Nodes are defined as an extension of the UML class concept. The Structure Nodes defined as an extension of the UML class concept are Window, Frame and Frameset (see Figure 3-26). The other two Structure Nodes (i.e. Presentation Page and Page chunk) are defined as an extension of the UML Package concept (see Figure 3-27). Each of these concepts is represented by means of a stereotyped class.

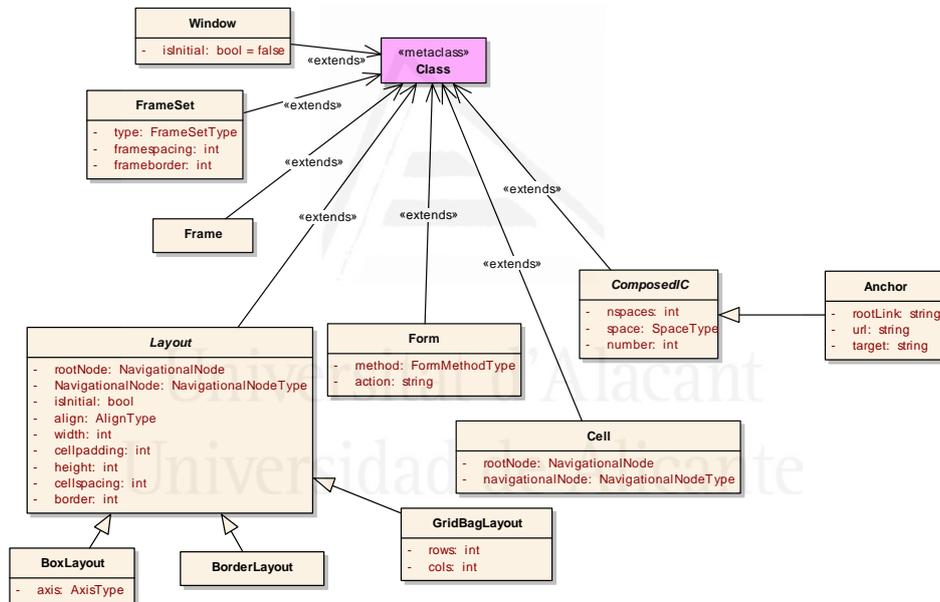


Figure 3-26: Presentation Nodes as an extension of the UML class concept

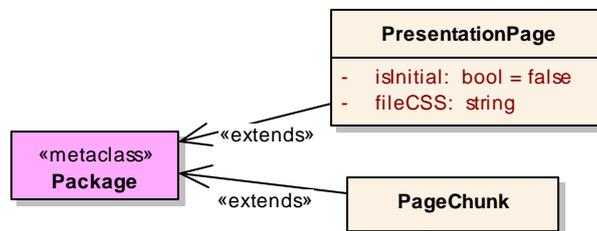


Figure 3-27: Structure Nodes

- The Interface Component *Anchor* also extends from the UML class concept, and it is represented with the stereotype <<Anchor>> (see Figure 3-26). It can contain Simple Interface Components which are represented by UML attributes (see Figure 3-28).
- Analogous to the Interface Components, Forms also extend from the UML class (see Figure 3-26) concept and its elements extend from the UML attributes concept (see Figure 3-28).

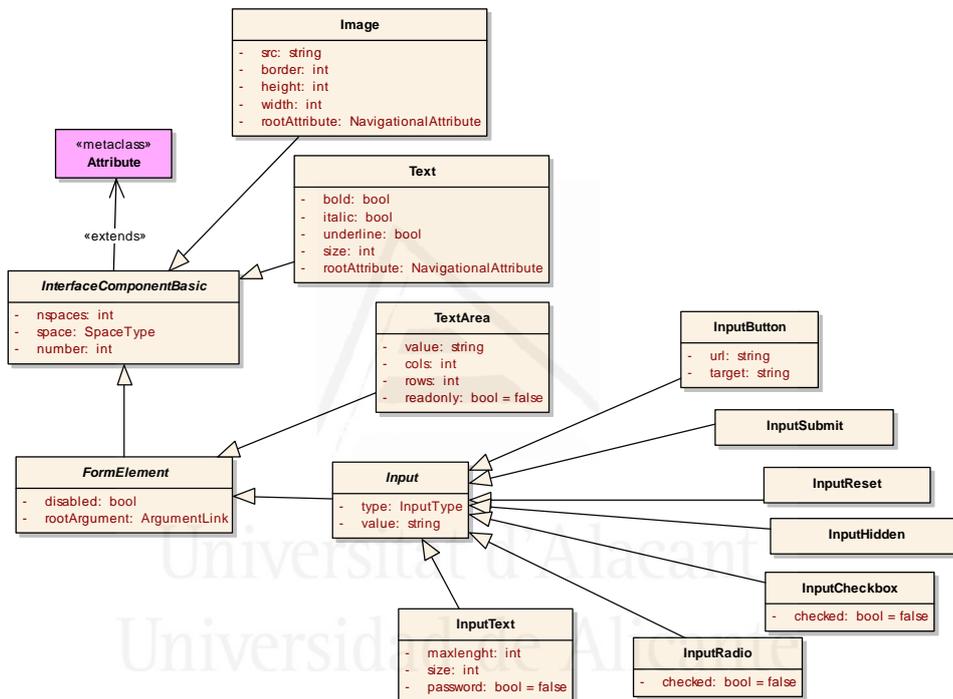


Figure 3-28: Interface Components and Form Elements

- The DPD Presentation Links *IncludeFrame* and *IncludeCell* are defined as an extension of the UML Association metaclass. The IncludeFrame link is represented with the stereotype <<IncludeFrame>> and the IncludeCell link with the stereotype <<IncludeCell>> as it can be seen in Figure 3-29.

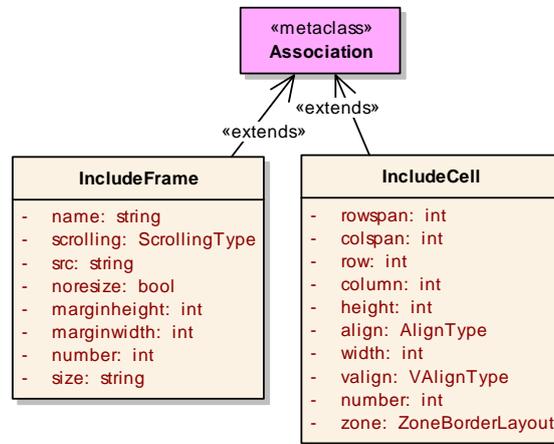


Figure 3-29: Associations

- The rest of the Presentation links (i.e. Navigates, Contains and Presents) are defined as an extension of the UML dependency metaclass and represented with the stereotype <<Navigates>>, <<Contains>> and <<Presents>> respectively as it can be seen in Figure 3-30.

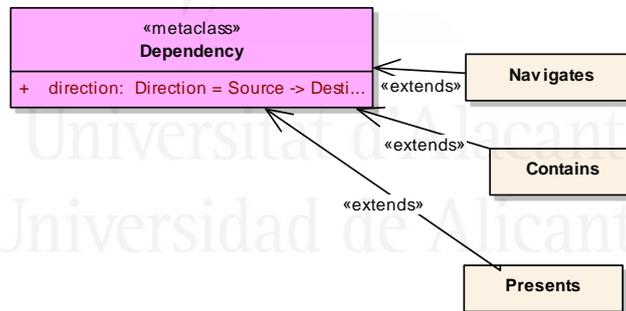


Figure 3-30: Dependencies

### 3.3.3.4 Example

For a better understanding of the DPD, in this section an example is shown based on a simplified virtual shop Website. The models are based in the NAD presented in Section 3.3.2.3.

The zero level of the DPD shows how the navigation nodes are grouped into abstract pages (and page chunks in the case) in the Website. In this case, each

of the Navigational Nodes correspond with one Presentation Page, excepting for the *Store* and *Category* classes that are grouped into one abstract page (i.e. the *Home* Presentation Class)

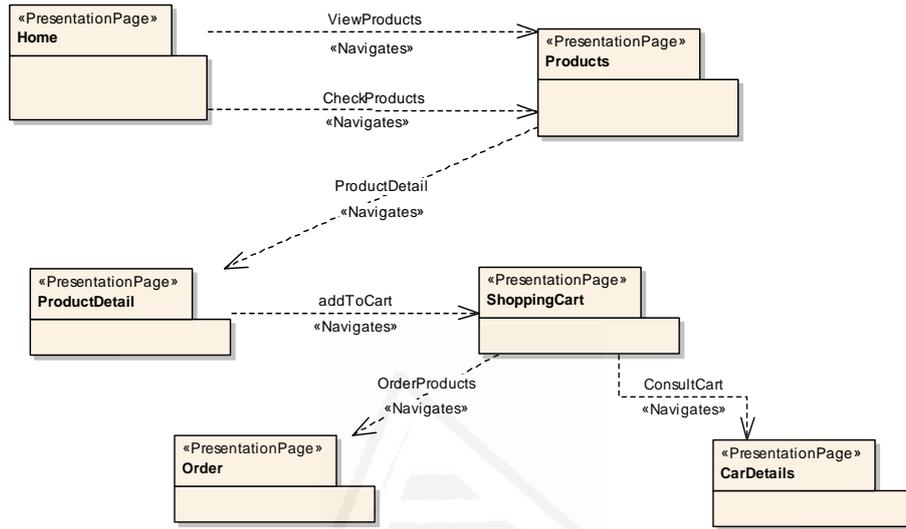


Figure 3-31: Level 0 of the DPM

When exploding each of the packages representing the pages, the elements composing them are defined. In Figure 3-32 the home presentation page detail is shown. This page is defined as a set of boxlayouts. The layouts are composed of cells which contain interface components that represent the elements of the Web page.

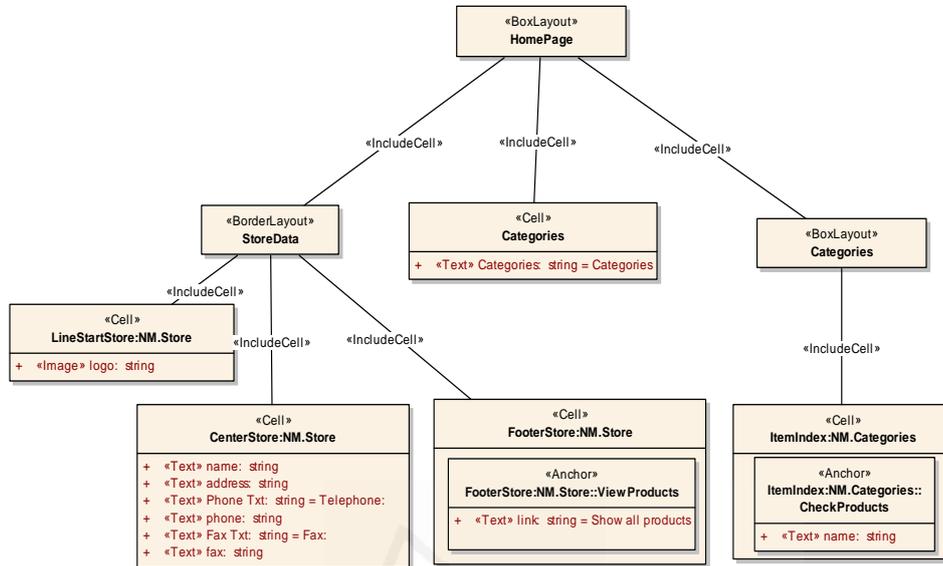


Figure 3-32: Level 1 of the DPM: Home Presentation Page

A possible representation of the final Web page is in Figure 3-33. The HomePage Boxlayout has three main cells.

- In the first of these cells a BorderLayout element is defined to place the information of the virtual shop. In the lineStart of the BorderLayout the logo of the shop is placed. In the center element of the BorderLayout the information stored is shown in form of textual interface components. In the footer element the link ViewProducts with the text “show all products” is placed.
- In the second main cell, the static text “Categories” is placed.
- In the third cell the different product categories are shown in a nested BoxLayout. Note that the categories had attached an Index pattern to be shown.

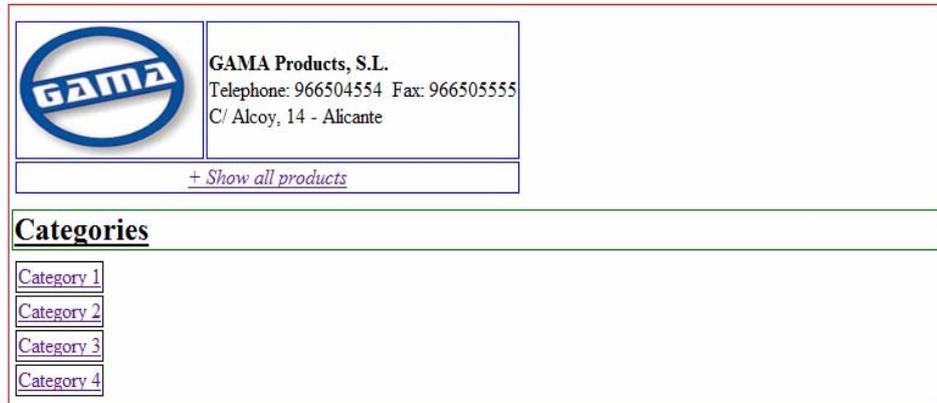


Figure 3-33: Possible representation for the Home abstract page

Once the Domain, Navigation and Presentation Models are defined, the Adaptation Design phase allows to define the adaptive behaviour of the Web application.

### 3.3.4 Adaptation Design

As aforementioned the OO-H method has been extended to support the modelling of adaptive Web applications. The goal is twofold:

- (1) provide the designer with the means of gathering, updating and storing all information needed to personalize the site, and
- (2) provide the designer with the means of specifying the personalization policy for the different users (adapting structure and/or content and/or presentation).

According to these goals, in the adaptation design workflow two models are defined: the *User Model* and the *Personalization Model*. We now discuss them in more detail and indicate how they are related.

#### 3.3.4.1 User Model

The user is one of the fundamental sources of information for the process of adaptation. In order to provide an individualized (i.e. personalized) Website, an adaptive system must have relevant knowledge about the user. The *User Model* (UM) specifies the structure of the personalization (including user)

data. The UM data should be defined based on the personalization requirements that we want to support in a concrete application. Typically these data (i.e. personalization data) is in some way related to the domain data. It is composed by descriptions that are considered relevant about the knowledge and user's aptitudes, providing information for the environment of the system for being adapted to the individual user. It can also contain domain independent traditional characteristics (like age and language) but also information related to the browsing behaviour of the user, as well as session information and information about the context in which the user is using the site. It also can contain user data referring to the domain (for example, information on the user's knowledge of topics occurring in the Website). This information builds the user profile and will be updated during the lifetime of the Website.

User models can be classified by different criteria:

- One criterion to distinguish different user models is *granularity* [Rich, 1983]. The scale ranges from a single model for all users (*canonical UM*), to models for each *individual* user. In between, there are models for groups of users with some common characteristics (*stereotypical UM*). In A-OOH the UM is defined for each *individual* user (for personalization support) but it can be also defined for user groups defining a *stereotypical UM* (of course adaptation for all users can be supported with a *canonical UM*).
- Another criterion is *temporal extent* of the data acquired during a session. If the data is only valid during a session or in the current context the data is considered *short-term data*. If the information is kept beyond the current session the information is classified as *long-term data*. In A-OOH the UM can contain *short-term* and *long-term* data of the user.
- Finally, the data can be represented by different types of models. If a *quantitative model* is used, the information stored is quantitative data like the amount of time the user has spent with the system. If a *qualitative model* is used, the information can be stored as an *overlay model* (the user's knowledge is stored as a subset of the knowledge of an expert) or as a *differential model* (they store the differences of the user's knowledge from that built into the system). In A-OOH qualitative and quantitative data can be stored. The UM is represented as a separate model that can store information about concepts of the domain (overlay model) and it can be represented as part of the DM. The A-OOH UM is defined as a

compliant UML class diagram. It is centered on the concepts of user and role (or user groups), the same as in other hypermedial approaches [Ceri et al, 2000]. The information we can store in the A-OOH UM can be generally classified into two types:

- *Domain dependent information*

The user information related to the domain is needed to allow personalizing the Web site and is domain dependent (e.g. history of buys in e-commerce sites, knowledge of the user in certain concepts in an e-learning scenario...).

- *Domain independent information* (user-specific data)

Besides the domain dependent information, there are other types of data (such as user browsing behaviour etc) that are features independent from the domain and dependent only on the user. This information can be classified into:

- *Independent on the session* (called *long term data* e.g. user age) or
- *Session dependent* (called *short term data* e.g. the device the user is browsing with in a concrete session).

*Qualitative data* can be stored as information independent of the session or session dependent, depending on the designer's goal (e.g. amount of time spent in the Website by the user in one session, or total amount).

In A-OOH the user-specific information is classified into four adaptivity dimensions (or criteria) upon which a personalization strategy can be built, namely *user characteristics*, *user requirements*, *user context* and *user browsing behaviour*. We can also define (dynamic) user-groups based on these criteria (as will be seen in next chapter). These criteria are explained next:

- *User Characteristics*

*User characteristics* (such as age or language) may be relevant for the structure, content or presentation of the Web site. Designers might try to exploit the user' characteristics to adapt the site to either arrive at a better usability and/or have a greater benefit for the company. Examples include: adapting the font size for users with low vision; avoiding the use of specific colours in case of colour blindness; offering shortcuts for experienced users; providing extra offers to wealthy customers; adding animations for young

visitors; offering links concerning the user's hobbies; etc.

- *User Requirements*

Every user comes to a Web site for specific reasons, and with specific goals. He is looking for some specific information or functionality and he expects to find it on the Web site, i.e. the user has user requirements. A good Web site design method starts out with specifying the user requirements, elaborating these requirements further and specifying the information or functionality needed to fulfil the requirements at a conceptual level (e.g. using modelling techniques such as UML, ER, ORM, ...).

Next, these conceptual representations are translated into actual Web pages, using some presentation and layout mapping. To be able to fulfil the requirements of a user, we will store information on the requirements in the user model. Tracking the user's browsing behaviour, we can detect which information and functionality he actually accesses. In this way we have knowledge on which requirements actually belong to the user and which don't, and we can adapt the site accordingly. As an example of adaptation based on (fulfilling) user requirements, we can consider adding a link to information relevant for a certain user, even though it was not originally assessed as a requirement for this particular user. Other examples could include adding a link on the homepage to the information/functionality (fulfilling a certain requirement) that was accessed the most, or removing content related to a requirement that was never accessed after a certain number of sessions.

- *User Context*

The variety of devices that can access the Web gets larger every day, and the particularities of the different devices may influence the way we want to structure or present information for/to the visitor, or even which information we offer. E.g. when the user has a small display available (e.g. mobile phone, PDA, mobile MP3 player) we don't want to overwhelm him with too many links or too much information, or when we know that the user is using a poor quality bandwidth connection, we want to avoid sending him large chunks of information (for example, media files). In other words, information concerning the context of the *current session* (i.e. this information is session dependent) of the user will be relevant to consider for personalization. In A-

OOH four kinds of context are considered: *LocationContext* (ubiquity of the user), *NetworkContext* (e.g. latency, speed, bandwidth ...), *DeviceContext* (e.g. PC, PDA, WAP ...), and *TimeContext* (date and local time of the connection).

- *User Browsing Behaviour*

The activity of the user in the system can comprise different aspects like: searches, visited pages, acquired products or services, etc. It is important to note that asking explicitly this information to the user by means of an input form is an invasive way of acquiring this data, and most of the times following this strategy the user does not personalize. It is then needed to acquire this information in a transparent way to the user.

The information stored in A-OOH is simple browsing behaviour of the user, this means the clicks of the user on a certain link can be detected and store information about it (e.g. number of clicks on a link).

To formalize the UM elements and their relationships the UM metamodel is defined next.

#### 3.3.4.2 UM Metamodel

Depending on the particular needs of the site or the domain covered, the designer should define a UM with correspondence with this MOF metamodel. (an example is given in Section 3.3.4.3).

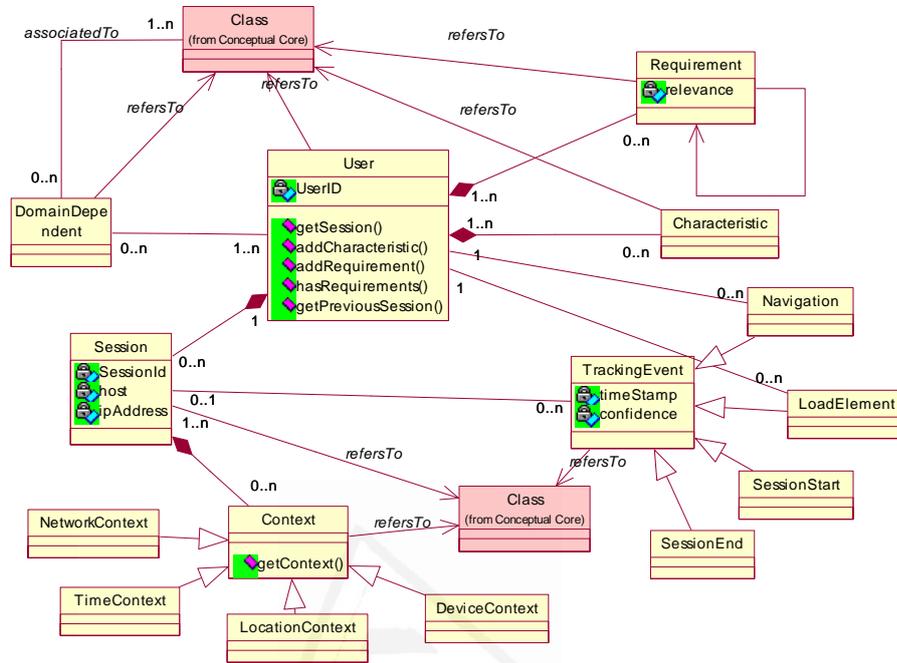


Figure 3-34: A-OOH UM Metamodel

In the A-OOH UM the user's characteristics and requirements are considered session independent information, and thus are associated directly to the user metaclass. The different types of context and the user behaviour actions are dependent on the session, they are associated to the user by means of the session metaclass. However, the navigation and loadElement events can also be stored as long-term data (this is, independent of the session) depending on the designer goals (e.g. to store the total amount of clicks on a certain link), that is why they are also directly associated to the User metaclass.

For each requirement the relevance to the user is stored (standard defined as the number of sessions in which it was accessed, relative to the total number of sessions) and possible logical connections to other requirements.

As we can see in the UM metamodel, the User class can be associated with DM classes as well as with domain dependent data (*DomainDependent* metaclass).

### ***Tracking Events***

In A-OOH when the user enters to the system, the User class of the UM is instantiated, that immediately causes the instantiation of a new Session object. If the user is registered in the system, this anonymous user is destroyed (and his corresponding session) and the corresponding object for the identified user is reactivated (persistent object). The system assigns then a new session to this user that is added to the (previous) set of sessions of the user in the system. Session objects store information about four different types of tracking events in the system<sup>3</sup>:

- *Start of Session*: it implies the entrance of the user in the system.
- *Navigation*: it implies the activation (by the user) of a navigational link.
- *Load Element*: it implies the load of an element (i.e. node) in the system.
- *End of Session*: it implies the exit of the user from the system.

All of these tracking events have associated a time stamp that determines the order in which the actions have been performed. The creation of new tracking objects and the association with the user's session it is done by means of the association of the services for the tracking event objects creation to the different types of browsing actions the user can perform:

- The activation of the entrance of the user in the system is implicitly associated with the creation of a tracking item of SessionStart type. When this event is triggered we could store the start page or link from which the user entered the application.
- The action of activating any navigational link in OO-H, besides to the navigation effect implies the creation of both a Navigation tracking object and a loadElement tracking object. The navigation object stores (besides the object representing the link) the information referring to the navigational context (i.e. origin and target of the link, possible filters, etc). The load element object stores the information referring to the loaded element. In this case we can store information in the UM like the number of clicks on a navigational link or the number of times the user visits certain node, etc.

---

<sup>3</sup> The A-OOH events will be explained in more depth in chapter 4.

- Finally, the instantiation of a tracking object of the type End Session (which causes the end of the session in the system) it is performed in an automatic way after certain inactivity of the user in the system (upper to a threshold, now set to 5 minutes). When the end event is triggered, we could store for example the last page the user visited in the application.

This information is useful to include the definition of several algorithms that can support different personalization strategies. An example can be defining a method “*User.getMostVisitedLinks()*”.

#### 3.3.4.3 UM Profile

In this section the profile defined for the UM is presented. The purpose of defining an UML profile is to provide an easy mechanism of adaptation to the UML metamodel to elements that are specific of a particular domain, platform or method. In this sense, the particular profile for the UM consists in adapting the elements defined in the UM to the UML metamodel. In this way the profile allows to specify the UM elements in any commercial tool that supports UML. The UML profile defined for the UM is in Figure 3-35.

- The different criteria considered (Characteristics, Requirements, Context) in the UM are defined as an extension of the *UML class concept* which has attributes and operations (also extensions of the UML concepts). There have been defined different stereotypes for representing the different types of criteria (i.e. <<Characteristic>>, <<Requirement>>, <<DeviceContext>>, <<TimeContext>>, <<LocationContext>>, <<NetworkContext>>) in the final model.
- The events are also defined as an extension of the *UML class concept*. The defined stereotypes for the different events are: <<SessionStart>>, <<SessionEnd>>, <<Navigation>>, <<LoadElement>>.
- Finally the Session is also defined extending the *UML class concept* with the stereotype <<Session>>.

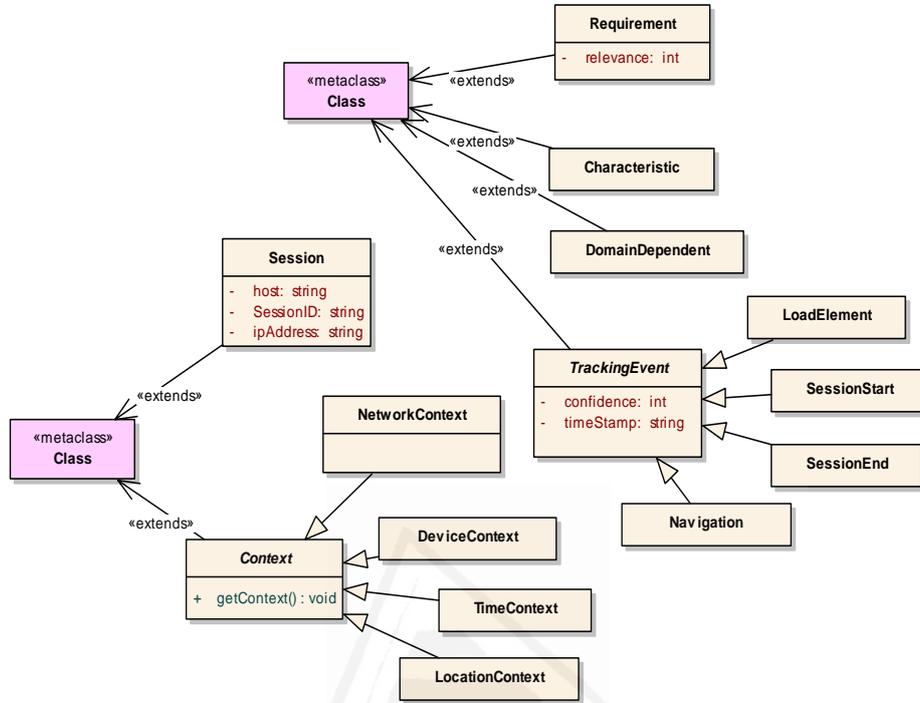


Figure 3-35 UM profile

### 3.3.4.4 UM Example

To build the UM we have to know which data we want to store. Let's see now an example of UM. Consider now an online library system in which we need to store/update the country the user is from (note that we are not referring to the country in which the user is browsing at the moment). This information is classified as a user characteristic and is stored in the UM adding a characteristic class, with the attributes needed to store.

Moreover we need to check if the user is browsing the Website with a PDA. For that we store the device the user is browsing with. This information is classified as the device context of the user and for storing it is needed to add a class stereotyped as "device context" to the *session* class.

Finally, the user has the requirement "ConsultBook". To store this in the UM we add a class stereotyped as "requirement" and we associate it to the *user* class.

Besides this domain independent data, some domain dependent information is stored: the user interest on the book's authors. And finally we store information about the browsing behaviour of the user: the number of times the user clicks on the navigational link "ViewDetails" during a session. In Figure 3-36 we can see the UM for this example.

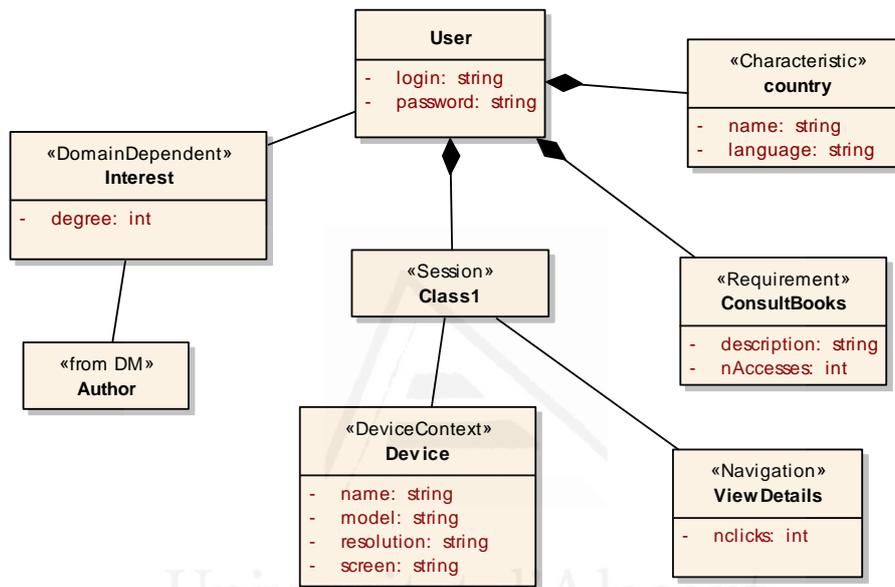


Figure 3-36: UM example

In this section the data structures for storing the needed information in the UM are shown, but how this information is acquired? Once acquired, what is done with this information? How is it used for personalizing the Website?. For answering these questions the personalization model is explained next.

#### 3.3.4.5 Personalization Model

The personalization model allows the designer to define a collection of rules that can be used to define a personalization strategy for a user or group of users. The rules are Event-Condition-Action [Ying et al, 2007; Heimrich and Specht, 2003; Chakravarthy, Le, and Dasari, 1999; Dayal, 1988] rules: they are triggered by a certain event (e.g. a browsing action, the start of

a session) and if a certain condition is fulfilled (for example “user.age=18”), the associated action is performed.

The rules will be defined using a simple and easy to learn language defined in A-OOH. One of the purposes of this language is to help the designers<sup>4</sup> to define all the rules required to implement a personalization strategy. It is a semiformal language defined with a BNF notation. This language is called PRML (*Personalization Rules Modelling Language*).

PRML was born in the context of the OO-H approach to extend it with personalization support. The main objectives of PRML are the following:

- **Specifying personalization at design time:** The main purpose of PRML is to provide the designer with an easy way of specifying adaptation actions at design time in A-OOH that will be performed at runtime.
- **Simple and easy to use:** Specifying adaptation at design time should use a simple and easy to learn formalism. The learning curve for the designer should be small. PRML is created as a *high level language* to abstract the designer from the programming code.
- **Enough expressiveness:** The language should be simple and easy to use by Web designers, but at the same time efficient having enough expressiveness to support simple and complex personalization policies specification. PRML should facilitate the specification of personalization policies such as the mapping from concrete personalization requirements to concrete PRML specifications (i.e. rules) is clear. Moreover the language should support the designer with a straightforward and easy way to understand the set of constructs supporting personalization techniques.
- **Separation of concerns:** The language should support personalization specification that is independent (as much as possible) from the rest of the models defining a Web application, mainly from the other application logic. To cope with this requirement PRML rules are defined as ECA (event-condition-action) rules. These rules can be separated well from the other functional specification of a Web application by means of independent event triggering and secondly, they represent a straightforward implementation of well-defined personalization policies.

---

<sup>4</sup> Personalization designers are not necessarily experienced Web programmers.

Therefore, PRML is defined as a *high level ECA rule-based* language allowing the designer to define easily simple and complex adaptation at design time (to be performed at runtime). However the fact that PRML was specifically created for the OO-H approach caused some syntactic and semantic problems:

- **Too Specific Syntax:** PRML was highly linked to the OO-H constructs. PRML syntax was too specific for OO-H so the rules were difficult to understand by people not familiarized with the OO-H approach. It also implied that PRML rules were difficult to compare with other specifications. To solve this problem, PRML syntax evolved to be *method independent*.
- **Restricted Semantic:** PRML semantic was restricted to the one supported by OO-H. Personalization actions were limited to the ones described in the OO-H approach. This problem caused to consider the following questions: *are the PRML constructs the “right” ones? Can PRML perform adequate set of personalization actions?*. To answer these questions PRML constructs were extended for being *method independent*. Making PRML *method independent* is not an easy task, it requires identifying a common denominator of a set of existing modelling language, abstraction the core set of concepts, and then developing a common rule language on top of the abstract concepts.

Making PRML method independent implies that it can be (re)used in different approaches. PRML objective has been not only being reusable but also supporting specific personalization (for different approaches). To cope with these requirements, two levels of PRML are defined:

- *PRML Lite*. This level allows the independence on concrete Web and data modelling methods, the reusability and for this purpose supports an adequate set of personalization actions. PRML Lite is explained in chapter 4.
- *PRML Full*. This level requirement is to give support for specific personalization, limiting though the reusability. With the constructs specified in this level more complex personalization strategies can be defined. PRML Full is explained in the chapter 5 of this dissertation.

PRML Lite is a subset of PRML Full, as we can see in Figure 3-37 so the particularities described in PRML Lite are also valid for PRML Full.

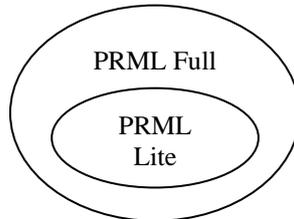


Figure 3-37: PRML Full and Lite

In both PRML Lite and Full, for satisfying a personalization requirement we have to define where and what information is acquired to obtain the required knowledge to personalize and define the personalization in terms of the effects this personalization causes in the system. For this purpose PRML supports the definition of two main types of rules:

- *Acquisition rules*, which help to acquire and store (at runtime) the information needed for personalization (for example to store the amount of accesses to a certain resource). This information is stored in the UM.
- *Personalization rules*, which describe adaptation actions (for example to state no colours are used if the user is color-blind). There are three kinds of personalization rules: *navigation rules* (to alter navigation), *content rules* (to add/remove/adapt content) and *presentation rules* (to modify presentation).

As aforementioned, the PRML language is explained in depth in chapters 4 and 5.

### 3.4 Implementation

Implementation is the following workflow considered in A-OOH. It is needed some data related with the implementation, which will be considered when generating the Web interface. The main activities in this stage are defining the *software architecture*, *integration* with the external data sources and Web services offered, *configuration of generation parameters* (such as client and

server languages, use of cookies... etc) and finally the *generation* of the final application.

### 3.5 Test

Finally in the last workflow, A-OOH includes two evaluation activities: a *prototyping* activity that allows simulating the structure and behaviour of the modelled interface without generating any line of code and a *validation* activity, done once generated the code of the interface. The details of these activities are outside the scope of this dissertation.

### 3.6 Conclusions

In this chapter we have presented the Adaptive OO-H (A-OOH) fundamentals describing its design process and the different diagrams considered. As a result of performing the activities of each of the design process phases, in A-OOH we get a set of models, reflecting views of the interface to be generated. A-OOH is an extension of the OO-H approach for supporting the modelling of adaptive (and personalized) Web applications. The main differences with respect to OO-H are next:

- Adaptive hypermedia systems are complex systems which require an appropriate software engineering process for their development. This is why in A-OOH the design process is based on the Unified Process (UP) and not in the spiral model as OO-H design process was based.
- The Navigational Model has been modified separating the presentation features that were mixed in the Navigational Model of OO-H. Moreover a UML profile has been defined for using UML notation for representing the Navigational Model.
- A Presentation Model has been added. This model also uses UML notation.
- A User and Personalization Model have been added for being able of modeling adaptive Web applications.

In the next chapters we introduce the Personalization Rules Modeling Language to specify a (reusable) personalization strategy. Concretely we present PRML Lite in Chapter 4 and PRML Full in Chapter 5.



Universitat d'Alacant  
Universidad de Alicante

## Chapter 4

### Personalization Rules Modeling Language

The previous chapter presented the A-OOH method and the extensions done to OO-H for supporting personalization. OO-H is extended to support personalization with two new models, namely the User Model (UM) where the information needed to personalize is stored (and updated), and the Personalization Model (PM) expressed as a set of rules. These rules define personalization policies for the Website. A personalization policy fulfils a personalization requirement expressing how and when the Website is going to be adapted (personalized) and it is defined with a set of rules.

Rules allow the decomposition of a personalization policy into single actions and this fact allows a transparent mapping from a policy description to particular rules. More specialized ECA (Event-Condition-Action) [Ying et al, 2007; Heimrich and Specht, 2003; Dayal,1988] rules represent a framework that allows rule triggering based on events that can be associated with particular actions that are taken by users when they are browsing a personalized Web application. ECA rules are therefore a convincing way for describing personalization policies - they can be separated well from the other functional specification of a Web application by means of independent event triggering and they represent a straightforward implementation of (well defined) personalization policies.

As explained in Chapter 1, Most of the existing methodologies allow developing Personalized Web Sites (PWS) where the personalization specifications are embedded in their design models. When the personalization is tightly intertwined to the rest of the application and the underlying technology, some problems arise like the difficulty of maintenance of the personalization and the inability of reusing the personalization specification among different approaches.

Our proposal treats the personalization as an orthogonal concern. Personalization is considered as a separate aspect of the Web site independent of the underlying technology, allowing the reuse of the personalization specifications. The proposed solution is based on a high-level rule language called PRML (Personalization Rules Modelling Language) [Garrigós et al, 2005; Garrigós and Gómez, 2006] that the designer can use to specify (at design time) the personalization to be performed at runtime. As personalization has become a key aspect in Web development, designers are often forced to add personalization to an existing Web site. Due to the separation of concerns, PRML also allows the designer to add personalization to an already generated Web site.

PRML was born in the context of the OO-H Web design method to extend it with personalization support. PRML evolved to be a generic language independent of the underlying technology. To achieve this independency it has been required to identify the common denominator of the most representative Web design methodologies, by abstracting the basic concepts. PRML has been developed using these abstract concepts. The double purpose of PRML is making possible the reuse of personalization strategies among different approaches and allowing the definition of more complex (and specific) personalization actions. For this purpose, two conformance levels are defined in this language:

- *PRML Lite*: This level comprises the basic operations supported by the most representative Web modelling methods [Casteleyn et al, 2003; Ceri et al, 2003; Gómez et al, 2001; Houben et al, 2004; Koch, 2001; Rossi et al, 2001]. It allows the independency from the Web modelling methodology and thus, the reuse. The purpose of its definition is being able to specify a (universal) reusable personalization policy.
- *PRML Full*: The purpose of this level is to give support for specific personalization actions, limiting though the reusability. With the constructs defined in this level more complex personalization strategies can be specified. PRML Lite is a subset of PRML Full; therefore the particularities defined in PRML Lite are also valid for PRML Full.

As already motivated in Chapter 1, we claim that, analogous to the figure of *presentation designer*, a role is needed of a designer responsible for defining the personalization on a Web site as an orthogonal aspect (i.e. *the personalization designer*). When defining a PWS, depending on the Web methodology used, the *personalization designer* should learn a different language for specifying personalization. This can be a cumbersome task for the designer, who does not have to be an experienced programmer. We argue that the designer can use PRML for specifying the personalization and then this specification is transformed to the specifics of the Web design method used. The *personalization designer* would only have to learn one efficient and easy to learn language, which is oriented to personalization.

PRML is a high level, simple and efficient language. It is a Domain Specific Language [Mernik et al, 2005] so the rules defined with it are very intuitive and easy to learn. DSLs allow solutions to be expressed in the idiom and at the level of abstraction of the problem domain. Consequently, domain experts themselves<sup>1</sup> can understand, validate, modify, and even develop DSL programs [Van Deursen et al, 2000]. As mentioned in the Conclusions Chapter, as future work we plan to do some tests with groups of users to check the usability of the language.

The learning curve of the *personalization designer* is reduced because we abstract the designer from the programming code and because the designer can define the personalization using PRML and then transformations to the specifics of each methodology can be done. Using PRML Lite we assure that the personalization specification can be mapped to any of the existing approaches, however specific mappings with PRML Full can also be defined, limiting though the reusability among methodologies. In this chapter we focus on PRML Lite, because as already stated this level is simpler and allows a higher reusability among the different existing approaches.

PRML is specified using BNF (Backus Naur Form) [Backus et al, 1960] Note that words in bold denote keywords and elements representing simple types (i.e. string, integer, date...) are not specified. For a complete specification of the BNF notation of PRML Lite consult the Appendix A of this dissertation and the full BNF specification of PRML Full can be found in the Appendix B.

---

<sup>1</sup> PRML users are supposed to be experts in the personalization domain.

In this Chapter we explain the fundamentals of PRML Lite. PRML Full is explained in Chapter 5. The outline of the chapter is as follows. The chapter begins presenting the case study used along the chapter. In Section 4.2 PRML Lite is explained in terms of the events supported, the way of defining conditions over the data of the UM, and the adaptation actions supported. Moreover is explained how to specify the execution order of the different PRML rules. Finally in Section 4.3 the conclusions of the chapter are given.

#### 4.1 Requirements for PRML Lite

PRML Lite's main purpose is that of making personalization specifications independent of the method and able to be portable to, or reusable by, other approaches. To cope with this purpose, PRML rules are defined as a separated component that can be connected or mapped to different Web modeling approaches. For defining PRML Lite the following steps were taken:

- *To identify a common denominator of a set of existing modeling languages and abstracting the core set of concepts*

To properly define a method-independent language, the first step is to identify a common denominator of a set of existing modeling languages and abstracting the core set of concepts. Different Web design methods supporting personalization were studied. The approaches considered for this study were WebML [Ceri et al, 2003], OOHDM [Rossi et al, 2001], UWE [Koch, 2001], Hera [Houben et al, 2004] and WSDM [Casteleyn et al, 2003]. This set of approaches is representative enough to show the generality of PRML Lite. All of them have personalization support except for WSDM, which currently supports adaptation for all the users of the website but could be easily extended to model personalization. Some essential requirements to be able to accommodate PRML Lite personalization specification were defined:

**R1:** A Domain Model (DM) should be a part of a Web application specification. The data structure of the website should have a way of being specified. The DM has to fulfill the following requirement:

**R1.1:** The DM should be composed of concepts, attributes and concepts relationships.

**R2:** A Navigation Model (NM) should be a part of a Web application specification. There should be a mechanism for accessing the data content of the navigation objects. This model has to fulfill the following requirement:

**R2.1:** The NM should have nodes and links as main components. Nodes are restricted views of domain concepts but can also be static nodes with no concept associated. Each node can have associated a concept from the DM or the UM. Nodes contain the attributes and operations that are going to be shown in the navigation.

Requirements R1 and R2 ensure that the target of the personalization rules exists and its data content can be properly referenced by condition and action parts of the rules. Domain and navigation specification are typically present in Web modeling approaches.

**R3:** A User Model (UM) should be a part of an application specification. An alternative can be a specification of updatable (temporary) data space. There should be a mechanism for accessing the data content of User Model objects.

Requirement R3 ensures that the target of the acquisition (data updating) rules exists and can be properly referenced. All modern methods also support some personalization or at least allow association of user actions with data updates.

**R3.1:** Analogous to the DM, the UM should also be composed of concepts, attributes and concepts relationships.

**R4:** Within the NM and UM, it should be possible to use events to trigger personalization, and conditions to execute the specific personalization policy.

Requirement R4 ensures that personalization can be triggered by events supported by the different methodologies. The personalization actions will be executed depending on the condition evaluation.

**R5:** Besides the previous requirements, it should be possible to define actions over the elements of the models.

Requirement R5 ensures that PRML actions have their counterparts in possible actions that can be specified using different methods.

The concrete issues we need to specify about these requirements (e.g. R4: which events should be supported, R5: which actions are required...) are

described along the next sections, while developing the common rule language on top of these abstract concepts.

- *Developing a common rule language on top of the abstract concepts*

Once having abstracted the models and concepts over which PRML has to operate, some aspects have to be defined to develop a common rule language:

- *Timing of the adaptation: when to personalize?* The events supported in PRML to trigger the rules should be defined.
- *Conditions.* The manner to refer to the concepts of the different models and the comparison operators supported in PRML conditions need to be specified.
- *Operations over concepts.* Finally, the operations allowed in PRML over the different concepts need to be described.

The development of the rule language regarding these issues will be presented in Section 4.3.

## **4.2 Case Study: an Online Library**

To better understand PRML, a case study is presented which describes an online library. In this library information about the books is shown, as well as reviews done by readers visiting our Website. Users can consult the books and buy them, adding them first to the shopping basket. In Figure 4-1 the Domain and User Model are shown. In the UM we store different information needed to fulfil the personalization requirements initially specified for the Website:

1. *Users that have bought at least 10 books will be offered a discount in the book price.*

In this case, to fulfil this requirement (and offer the price discount to the user) the number of books bought (in all the sessions) should be stored in the UM.

2. *Users will see recommendations of books in which authors they are interested in.*

3. If the user does not have enough interest in any book author to get personalized recommendations, the link recommendations is not shown

In order to fulfil the 2<sup>nd</sup> and 3<sup>rd</sup> requirements we need to acquire and update the user interest in the different authors.

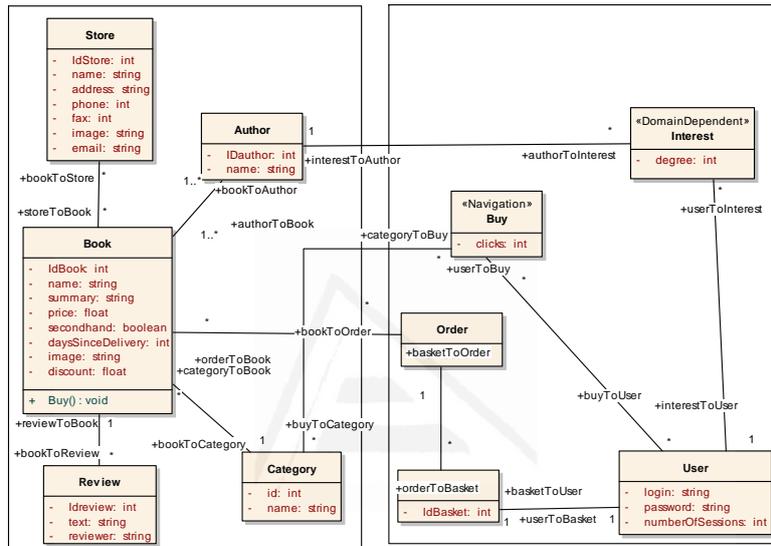


Figure 4-1: Domain and User Model for the Online Library Case Study

In the updatable data space defined by UM the information describing the user’s interests on the authors is stored as *DomainDependent* information, according to the defined personalization requirements. In the UM we also have the *Buy* class. This class represents a *navigation event* triggered by the user behaviour and stores the number of clicks done in the link *Buy* in the attribute *clicks* (we will use this information to fulfil the 1<sup>st</sup> personalization requirement of the running example). This value is stored as *long-term data* (i.e. independent of the session), because the designer wants to personalize on basis of the number of books bought in total by the user. Note that to store the number of clicks on any other link we would just have to add a new *Navigation* element to the UM.

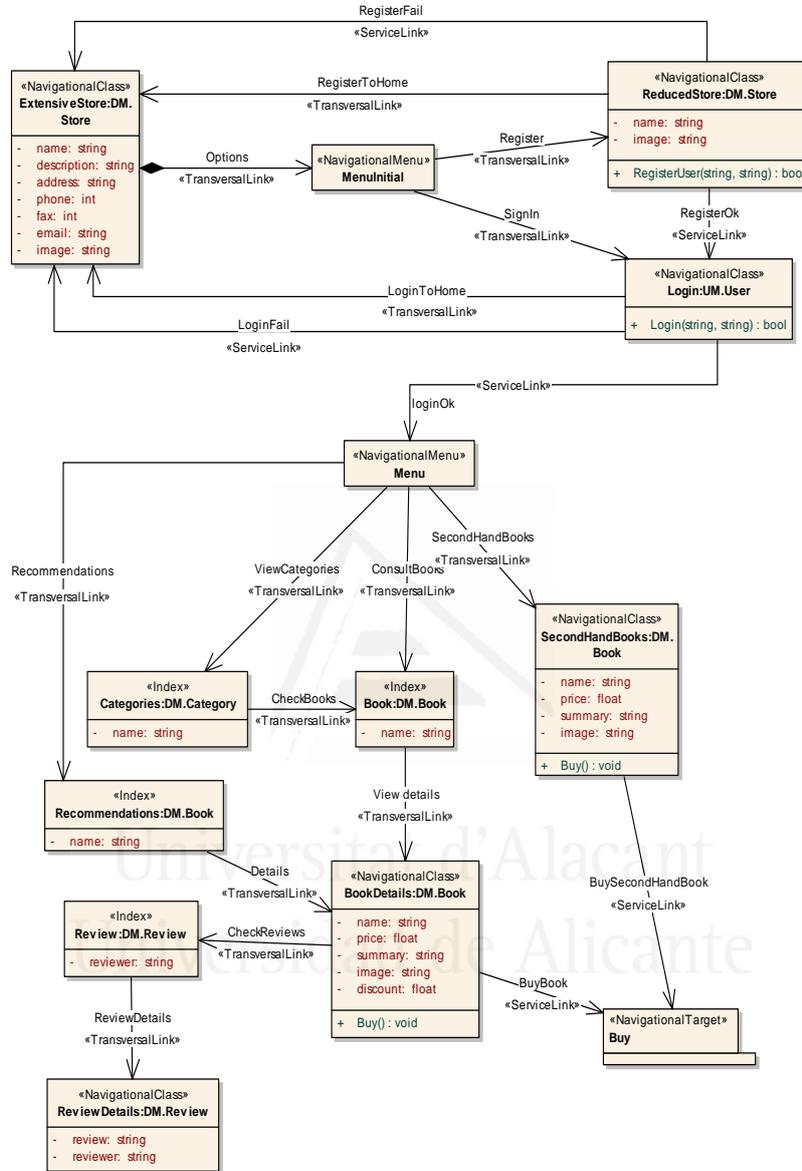


Figure 4-2: Navigation Model for the Online Library Case Study

In Figure 4-2 the NM of the online library is shown. When the user enters the Website he finds a collection of links (i.e. menu) with the links *ConsultBooks*, *Recommendations*, *ViewCategories* and *SecondHandBooks* as a starting point. If the user navigates through the first link (*ConsultBooks*) he will find an

indexed list of all the books (indexed by the book's name). The user can click in any of the book names to view the details of the chosen book (see in Figure 4-2 the navigational class *Book:BookDetails*). Moreover the user can see reviews done by other users of the different books. When the user clicks on *ViewCategories*, an indexed list of the categories is shown (indexed by the category's name). When the user clicks in one of the categories he will see the books associated to that category. If the user navigates through the *SecondHandBooks* link he can see all the books used that are on sale. When the user clicks on *Recommendations*, personalized book suggestions should appear (based on information from the User Model).

For fulfilling the personalization requirements we still need to specify how to update the information in the User Model and we have to specify the personalization actions needed. This is specified in the Personalization Model, that is expressed as a set of personalization rules. As aforementioned, these rules are expressed *using* the PRML language. For the specified requirements we only need the PRML Lite, which is explained along this chapter.

PRML Lite is based on a MOF [OMG Meta Object Facility] metamodel (see Figure 4-3) which defines the set of constructs of the language such as the different parts that form a PRML Lite rule and the different events and the actions supported. The main element of the metamodel is the rule metaclass which represents the concept of rule containing the elements that define it. The elements defining a rule are the ones that represent its main structure and are explained along the chapter.

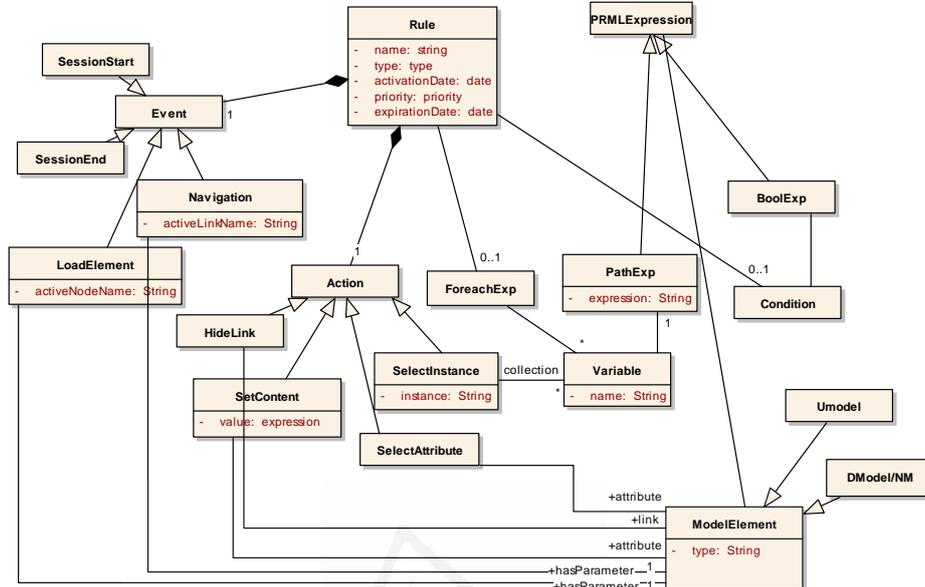


Figure 4-3: (Simplified) PRML Lite Metamodel

The basic structure of a rule defined with PRML is the following:

```

When event do
    [Foreach expression]
    If condition then action [else action] endif
    [endForeach]
endWhen
    
```

A PRML rule is formed by an event and the body, which contains a condition (optional) and an action to be performed. The *event* part of the rule states when the rule should be triggered. Once triggered the rule *condition* is evaluated (if any) and in the case the *condition* is evaluated positively the *action* is performed. A *Foreach expression* can be also present in the rule when the action and the condition act over a set of instances. The condition may also have an *else* statement.

For satisfying a personalization requirement we often have to define how to update the knowledge about user (acquisition rule). This information is stored in the UM data space. Furthermore, we have to define the effects this personalization causes to the presented content and navigation structure

(personalization rules). These rules use the information specified by the UM to describe adaptation actions. Depending on the object of the adaptation two kinds of personalization rules are considered: navigation personalization rules (to alter navigation), and content personalization rules (to add/remove/adapt content).

In the next section we introduce the fundamentals of the Personalization Rules Modeling Language (Lite) to specify a personalization strategy.

### 4.3 PRML Lite Fundamentals

Once having abstracted the models and concepts over which PRML has to operate, some aspects have to be defined to develop a common rule language:

- *Timing of the adaptation: when to personalize?* The events supported in PRML to trigger the rules should be defined.
- *Conditions.* The manner to refer to the concepts of the different models as well as the comparison operators supported in PRML conditions need to be specified.
- *Operations over concepts.* Finally, the operations allowed in PRML over the different concepts need to be described.

These issues are described in next sections.

#### 4.3.1 Timing of the Adaptation

To specify when an adaptation (personalization) action should be performed, PRML allows specifying the following type of *user browsing events*:

- PRML Lite includes browsing events relevant for a website including its browsing sessions. We argue that the majority of mature Web systems and design methods distinguish the following browsing events: *link navigation, node loading, starting a browsing session, and ending a browsing session*. Browsing events are implicitly associated with the user clicks and may trigger the activation of one or more rules. PRML considers the following user browsing events:
  - *Navigation event:* caused by the activation of a link in the NM.

- *LoadElement event*: it is caused by the instantiation of a node of the NM. The difference between this event and the previous one is that the *LoadElement* event is triggered independently of the link that loads the node<sup>2</sup>. This is useful when a set of links have the same target node and we want to personalize during the activation of any of those links.
- *Start event*: triggered with the start of the browsing session by a user.
- *End event*: triggered when the browsing session expires after certain inactivity of the user in the system, or when the user explicitly finishes the session.

In the following section, the specification of these events in PRML rules will be explained.

#### 4.3.1.1 Events: PRML Specification

In the case of user browsing events which imply navigation of the user within the website<sup>3</sup> (i.e. *Navigation* and *LoadElement* events) information about navigation is passed as a parameter. As aforementioned, nodes are restricted views over the DM or UM concepts. The parameter contains information about which instances of concepts of the DM/UM are to be shown in the activated node. The instances to be shown in the nodes are adapted to each user (i.e. personalized) by means of PRML rules. Two different types of parameters are possible:

- *Simple parameter*: The event passes a simple parameter when in the target node an instance (always one) of a concept of the DM/UM is shown. The representation in PRML of these types of events transferring a simple parameter is:

```
When Navigation.activeLink (NM.activeConcept4 parametername) do
When LoadElement.activeNode (NM.activeConcept parametername) do
```

---

<sup>2</sup> Note that PRML rules can be attached to nodes or to links of the Navigation Model

<sup>3</sup> Excluding the browsing session events.

<sup>4</sup> To access a certain element of a model, PRML navigates over the model using path expressions (PE), see section 4.32.1 for details,

- *Complex parameter*: This parameter refers to a set of instances (one or more instances) of a concept of the DM/UM. The representation in PRML of these types of events transferring a complex parameter is:

```
When Navigation.activeLink (NM.activeConcept* parametername) do
When LoadElement.activeNode (NM.activeConcept* parametername) do
```

Every time the user activates a link (*activeLink*) a node is activated (*activeNode*) and the associated event is launched, which may trigger one or more rules. The parameter is the instance or set of instances from the root concept of the DM/UM (*activeConcept*) to be shown in the *activeNode*. There is an exception: when the Navigation event is triggered as the result of the activation of a link which invokes a service, the parameter passed is the instance or set of instances of the source node.

The distinction into simple and complex parameters is carried out to improve the readability and maintenance of the rules for the designer. In other words, if a complex parameter is transferred then a loop will be needed to go over the set of instances.

In the case of the running example, for specifying that an event should be triggered when the user clicks on the *ViewDetails* link in PRML, we express it as:

```
When Navigation.ViewDetails (NM.Book book) do
```

This event transfers a simple parameter (an instance of the book visited). An example of transferring a complex parameter is shown below:

```
When Navigation.Recommendations (NM.Book* books) do
```

When the user clicks on *Recommendations*, a set of books (zero or more) will appear (with no adaptation action applied). An appropriate PRML rule will adapt the set of books for a particular user (explained in Section 5.3.3).

The browsing session events (i.e. *Start* and *End* events) do not transfer any parameter. Their PRML representation is very simple:

```
When SessionStart do
When SessionEnd do
```

The BNF specification for the PRML events can be found in the appendix A of this dissertation.

### 4.3.2 Conditions in PRML Rules

PRML offers a high degree of freedom in defining *conditions*, while it is still simple and minimal enough to be used in multiple methodologies. When specifying conditions two aspects have to be decided:

1. *How to refer to the different elements of the models defined in the website*
2. *How to specify conditions over these elements?*

#### 4.3.2.1 References to the Models in PRML Conditions

When specifying conditions, PRML rules can refer to different elements of the models defined for the website. As already explained, personalization is based on the user's data stored in the UM. This information can be updated and consulted (specifying conditions over those data to base the personalization on). For updating this information adequate actions are defined (see Section 4.3.3.1). Nevertheless, a mechanism to access the UM structures is needed. Besides, a PRML rule can also refer to information from the DM and from the NM for specifying some needed conditions to define adaptation actions. To indicate the access to data of a particular model the prefixes "DM", "UM" or "NM" are added in PRML rules to the expressions referring to the particular information. In the case of accessing information defined in the UM, the source concept is always "User", to identify the user that is actually browsing the website.

To access a certain element of a model, PRML *navigates* over the model using path expressions (PE). The PEs are based on the path expressions defined in OCL [UML 2.0 OCL specification]. As an example for a PE, consider that we want to access the interest degree of the user in the different authors stored in the UM (see Figure 4-1). For building the PE the following steps are to be taken:

- As already mentioned, the PE starts with a prefix indicating the model that is being *navigated*.
  - *E.g. UM*
- After that, it follows a dot and then the source concept which the expression departs from.
  - *E.g. UM.User*

- For traversing the concepts, the target roles of the concepts relationships are added to the PE (with dots).
  - *E.g.* `UM.User.userToInterest`
- It finishes with either a concept or an attribute of a concept.
  - *E.g.* `UM.User.userToInterest.degree`

When accessing information defined in the NM instead of concepts PEs refer to nodes, and instead of concepts relationships PEs refer to links. The BNF specification for a PE is described in the appendix A of this dissertation.

#### 4.3.2.2 Boolean Expressions: PRML Specification

PRML conditions over data use the PEs explained in the previous section to access the information of the different models (i.e. DM, UM, NM). The following operators and structures may appear in a PRML condition:

- Relational operators (`>`, `<`, `>=`, `<=`, `=`, `!=`)

```
if (UM.User.userToInterest.degree > = 100) then
```

- Arithmetic operators (`+`, `-`, `/`, `*`)

```
if (UM.User.userToInterest.degree + 20 > 140) then
```

- Logical operators (and /or)

```
if (UM.User.userToInterest.degree > = 100 and
UM.User.userToInterest.degree < 200) then
```

- Negation (not)

```
if not (UM.User.userToInterest.degree > = 100) then
```

- In operator (in)

```
if (UM.User.userToInterest.interestToAuthor in
books.booksToAuthor) then
```

This kind of condition checks if a certain instance is contained in a collection of instances.

- *ForAll Conditions*

```
If ForAll(UM.User.usertoInterest)
  (UM.User.usertoInterest.degree=null or
  UM.User.usertoInterest.degree<100) then
```

The ForAll condition is a special kind of condition: it is checked for all instances of the specified concept. This can be useful when we need to evaluate the condition for a set of instances but we want to perform the action only once (i.e. when the condition holds for all instances).

#### 4.3.3 Actions Supported

PRML Lite contains a set of constructs to model the personalization of a website manipulating the UM and NM. These constructs represent personalization actions over the different elements of those models (note that the elements of the different models were explained at the beginning of Section 4.2). As aforementioned, actions supported by PRML Lite have been abstracted from a selection of well-known modeling approaches. The adaptation actions supported by the studied approaches include the user data manipulation. All approaches allow filtering the content to show and most of them (UWE, Hera and OOHDM) allow filtering the links as well. In Table 4-1, we can see which actions are supported by each of the studied methodologies. We have selected the most common actions for PRML Lite.

Table 4-1: Actions Supported by the different methodologies

<i>Action / Methodology</i>	<i>UWE</i>	<i>Hera</i>	<i>WebML</i>	<i>OOHDM</i>	<i>WSDM</i>
<b>Updating UM Content</b>	Yes	Yes	Yes	Yes	No
<b>Filtering content (concept instances)</b>	Yes	Yes	Yes	Yes	Yes
<b>Filtering content (concept attributes)</b>	Yes	Yes	No	No	Yes
<b>Link hiding</b>	Yes	Yes	No	Yes	Yes
<b>Sorting content</b>	No	No	No	No	No
<b>Sorting links</b>	Yes	No	No	No	No

<b>Adding / deleting links</b>	No	No / No	Yes / Yes	No	Yes
<b>Adding /deleting concepts</b>	No	No	Yes / Yes	No	No
<b>Dynamically grouping users</b>	No	No	No	No	No
<b>Link annotation</b>	Yes	No	No	No	No

After this study we can conclude that this level of the language should contain the following set of operations over the UM and NM elements:

- **Actions over attributes (UM and NM):**
  - *Updating an attribute value from the UM (setContent):* This action allows modifying/setting the value of an attribute of a concept of the UM.
  - *Filtering attributes in the NM nodes (selectAttribute):* By means of this action a node can be restricted by hiding or showing some of the attributes of the DM/UM related concept.
- **Actions over nodes (NM):**
  - *Filtering node instances (selectInstance):* This action allows us to show only the selected instances of a DM/UM concept for a user depending on the personalization requirements that we are aimed at supporting.
- **Actions over links (NM):**
  - *Hiding links and their target nodes (hideLink):* Analogous to filtering data content, PRML also supports filtering links.

However, some adaptive operations not supported by most of the studied methodologies (and thus not supported by PRML Lite) are also interesting for personalization actions. These actions are defined in PRML Full (explained in Chapter 5):

- **Actions over nodes (NM):** In PRML Full, it is possible to sort node instances by a certain value to satisfy a personalization requirement.
- **Actions over links (NM):** PRML Full supports adding new links to the NM and deleting and sorting links.

- **Actions over users (UM):** PRML Full also allows to dynamically grouping users by certain criteria to attach personalization rules to them.

As we have already explained, in this chapter we will focus on the PRML Lite level, elaborating the different actions by means of the presented running example .

#### 4.3.3.1 Updating a value (*setContent*)

This action changes the value of an UM attribute. Consider the scenario presented in the running example in Section 4.1 of an online library. The acquisition of the information that needs to be stored in the UM is carried out by means of acquisition rules which perform the *setContent* action.

- To fulfill the 1<sup>st</sup> requirement, the number of books bought by a user has to be stored in the UM. The following rule updates the number of books bought when the user clicks on the *Buy* link.

```
When Navigation.Buy (NM.Book b) do
    setContent(UM.User.userToBuy.Clicks,
               UM.User.userToBuy.Clicks + 1)
endWhen
```

This rule is triggered by a *Navigation event* when the user activates the link *Buy*. It updates the number of books bought when the user clicks on the *Buy* link (updating the attribute *Clicks* of the *Buy* concept of the UM) using the **PRML setContent action**.

- To cope with the 2<sup>nd</sup> requirement the user's interest in the authors of the books has to be gathered. In this example, the interest of the user is defined in the following way: the user is interested in an author when he consults a book of him.

```
When Navigation.ViewDetails (NM.Book book) do
  ForEach a, b in (UM.User.userToInterest, book.bookToAuthor)
    If(b.ID = a.interestToAuthor.ID) then
      setContent(a.degree, a.degree+10)
    endIf
  endForEach
endWhen
```

This rule is triggered by a *Navigation event* when the user activates the link *ViewDetails*. It updates the interest of the user in the author of the visited book (by updating the attribute *Interest.degree* of the UM).

The *foreach expression* of the rule is needed to go over each of the instances of the *Interest* concept (referenced by the Path Expression *UM.User.userToInterest*) in the UM (see Figure 4-1) in order to check the proper instance to update. The *foreach expression* also goes over the set of authors of the consulted book. If the instance of the author checked in the UM is the same as one of the authors of the book the user is consulting when clicking the *ViewDetails* link then we will update the interest degree in that author in the UM, using the **PRML setContent action**.

#### 4.3.3.2 Filtering attributes in the NM nodes (*selectAttribute*)

To illustrate the *selectAttribute* action of PRML Lite, we consider the 2<sup>nd</sup> personalization requirement: When the user clicks on the *ViewDetails* link to consult a specific book, if the number of books bought is greater than 10, we will show an attribute “*discount*” to the user (note that this attribute was not previously selected to be shown in the Navigation Model but is present in the Domain Model).

```

When Navigation.ViewDetails(NM.Book book) do
  If UM.User.userToBuy.Clicks >10 then
    book.Attributes.selectAttribute(discount)
  endIf
endWhen

```

In this rule from the set of attributes of a book, the attribute *discount* is selected. The rule is triggered by a *Navigation event*, result of clicking on the *ViewDetails* link. Events pass a parameter with the information of the node to be shown, in this case the concrete book of which we are going to see the details. If the number of clicks on the *Buy* link (stored in the User Model with a rule explained in the previous section) is greater than 10, we will select the attribute *discount* from the book concept to be shown using the **PRML selectAttribute action**.

#### 4.3.3.3 Filtering node instances (*selectInstance*):

Let's consider now the 2<sup>nd</sup> requirement of the case study in which recommended books based on the interest of the user are shown. The interest of the user needs to be stored in the UM with an acquisition rule taking into account the user's browsing behaviour. This rule has been explained in Section 4.2.3.1. The personalization rule that displays the book names if the user has an interest degree in the book author greater than 100 is:

```

When Navigation.Recommendations(NM.Book* books) do
  Foreach b,a in (books,UM.User.usertoInterest) do
    If (a.degree > 100 and
      a.interestToAuthor.ID in b.booksToAuthor.ID) then
      books.selectInstance(b)
    endIf
  endForeach
endWhen

```

The rule is triggered by a *Navigation event*, result of clicking on the *Recommendations* link. As aforementioned, navigation events pass a parameter with the information of the node to be shown, in this case the set of all the books which is filtered to only show the recommended books.

In this example a *foreach expression* is needed to go over all instances of the set of books passed as an event parameter, and the loop also goes over all instances of the author concept. If the user has an interest degree greater than 100 related to the instance of the author we are checking, then the book instance checked is selected to be shown using the **PRML selectInstance action**.

#### 4.3.3.4 Link Hiding (*hideLink*)

To illustrate the *hideLink* action of PRML Lite, we consider the 3<sup>rd</sup> personalization requirement: If the user interest degree in the different authors is not initialized (this means that the user has not consulted the details of any book yet, maybe because it is the first time the user goes into the website) the *recommendations* link is not shown. Note that there is only one *Recommendations* link in the website for each user.

```

When SessionStart do
  If ForAll(UM.User.usertoInterest)

```

```

    (UM.User.usertoInterest.degree=null or
    UM.User.usertoInterest.degree<100) then
        hideLink(NM.Recommendations)
    endIf
endWhen

```

The rule is triggered by a *SessionStart event*, when the user enters the website. In this case, our purpose is to check a condition over a set of instances of a concept as well as to perform an action (hide the link) only if the condition is true for all the evaluated instances.. For this purpose the *Forall* statement is added to the condition stating the collection over which the condition should be true. This condition checks if the interest degree of the user in the different authors has any value. If it has no value, it means that the *Recommendations* link should not be shown, so it is hidden by using the **PRML hideLink action**.

For clarity reasons the rules specified in this chapter have been simplified by omitting some rule features (e.g. rule name, priority, activation and expiration dates, etc). They will be explained in the following section. Furthermore, in Section 4.2.5 we will explain how to specify the execution order of the different rules.

#### 4.3.4 Rules Features

When specifying a PRML rule we can state a set of rule features like the activation date on which the rule is first triggered, the expiration date on which the rule becomes inactive (and thus not anymore triggered), the name and the type of the rule and the *priority* over other rules. An example of such features specification is:

```

Rule: suggestions type: Pers-Content priority: high activation: 22-10-2005 expiration: 22-10-2006

```

This rule is called “suggestions”, the type of the rule is *Personalization-Content*. It has an activation date (22-10-2005) on which the user is first time triggered, and an expiration date (22-10-2006) on which the rule becomes inactive and thus not triggered anymore. The order of execution for rules sharing the same triggering event is unspecified unless it is enforced using

priority. This rule has a *high* priority which means that will be executed earlier than rules having *medium* and *low* priority.

#### 4.3.5 Execution Order

When a set of rules are triggered by the same event and they become active (i.e. their condition evaluation is true), all the active rules are collected in a queue (stack) and executed sequentially. Parallel execution would require a conflict resolution mechanism in case of several active rules performing conflicting actions. The execution of a rule may activate or deactivate other rules, so other active rules in the queue may become inactive before they are *selected* for execution. A different execution order of the rules in the queue can affect the final state (those rules will have execution order-dependence). For the same user the system should provide the same result; the same final state, this is called *confluence*. To decide the execution order of the active rules a priority parameter is defined, however if the order-dependent set of rules is greater than 3 (there are only three priority values: *high*, *medium*, *low*) we need another mechanism for assuring the *confluence*. Usually the order of execution by the rule engine is unspecified and depends on the specific rule engine used, but some constraints are defined:

- *Acquisition rules* have always higher priority than *personalization rules*.
- *Personalization rules* triggered by the same event affecting the same element should be defined taking into account that:
  - Filtering attributes has higher priority than filtering instances.
  - The order of execution of two rules with the same action type (over the same element) should be specified by means of the priority feature.
- Multiple *acquisition rules* sharing the same event should be avoided. The reason is that when unsynchronized data updates take place can lead to possible data inconsistency.

So far, we have presented the PRML Lite language. From now on, we will present how the transformations from PRML into UWE and Hera approaches are possible. The aim of the transformations is that of showing the portability

of the PRML approach, and thus the reusability of a PRML specification among different methodologies.

#### 4.4 Conclusions

Web site modelling methodologies provide a conceptual and systematic approach to (complex) Web application design and implementation. Most of these methodologies allow developing Personalized Web sites where the personalization specifications are embedded in their design models. This can cause some problems like the difficulty of maintenance of the personalization and the inability of reusing the personalization specification among different approaches. In this Chapter we have presented a solution based on a high-level rule language called PRML (Personalization Rules Modeling Language) which allows to specify the personalization at design time as an orthogonal concern of the Web site, independent of the underlying technology.

Moreover the fact of considering Personalization as a full discipline present in many fields besides Web Engineering asks for the role of a *designer* who is responsible of defining the personalization and who knowledge of psychology and sociology. In consequence, the *personalization designer* does not have to be an experienced programmer. The *personalization designer* finds the problem of having as many ways of specifying the personalization as the number of existing Web methodologies which support it. Depending on the Web methodology used, the *personalization designer* should learn a different language for specifying personalization. This can be a cumbersome task for the designer, who, as aforementioned, does not have to be an experienced programmer. We argue that the designer can use PRML for specifying the personalization and then this specification is transformed to the specifics of the Web design method used (such transformations are explained in the Chapters 7 and 8). This would reduce the learning curve of the personalization designer.

The double purpose of PRML is making possible the reuse of personalization strategies among different approaches and allowing the definition of more complex (and specific) personalization actions. For this purpose, two

conformance levels are defined in this language: PRML Lite and PRML Full. This chapter presented the fundamentals of PRML Lite. In the next chapter PRML Full is described.



Universitat d'Alacant  
Universidad de Alicante

## Chapter 5

### PRML Full

The Personalization Rules Modeling Language (i.e. PRML) defines two conformance levels: PRML Lite (presented in previous chapter) which is defined abstracting the basic (adaptivity) concepts of the most representative Web methodologies, and PRML Full which purpose is to support more complex and specific personalization actions limiting thus the reusability achieved with PRML Lite.

This Chapter presents the basics of PRML Full extending the case study used in previous chapter. The outline of the chapter is as follows: in Section 5.1, in order to illustrate the PRML Full possibilities, the case study defined in Chapter 4 is extended with more complex personalization requirements. Section 5.2 presents the fundamentals of PRML Full, presenting its metamodel and explaining the support of complex user browsing behaviour and the different adaptation actions supported by the language. Moreover some guidelines are given to the designer for properly defining adaptivity rules with PRML Full. Finally, in Section 5.3 the conclusions of the chapter are given.

As explained in Chapter 4 PRML is specified using BNF (Backus Naur Form) [Backus et al, 1960]. Note that words in bold denote keywords and elements representing simple types (i.e. string, integer, date...) are not specified. For a complete specification of the BNF notation of PRML Full consult the appendix B of this dissertation.

#### 5.1 Extension of the Case Study

In order to illustrate the different PRML Full actions, the case study describing an online library presented in Chapter 4 (Section 4.1) is extended with the following personalization requirements:

1. *Users will see the books sorted by their interest in the books' authors.*

This requirement states that the books will be shown sorted by the interest degree (in books' authors) of the user. To fulfil this requirement we have to store and update in the UM the user interest in the different authors.

2. *If the user has bought at least 10 books of the same category a new link is added to the homepage pointing to the new books on that category.*

In this case, to fulfil this requirement (and add the new link) the number of books bought (along all the sessions) on the different categories should be stored in the UM.

3. *The link SecondHandBooks is deleted when the user has not consulted it (more than once) during 10 sessions.*

To fulfil this requirement, the clicks done (per session) in the SecondHandBooks link are stored in the UM.

4. *If the user is less than 18 years old, the user is attached to the group "Minors". For the minors the following requirement is applied: books with the category "Terror" are not shown*

To cope with this requirement, we need to store the age of the user in the UM.

5. *If the user is browsing the site with a small screen device (i.e. PDA, MP3 or mobile phone) s/he is attached to the group "SmallScreenDevice" and for the users belonging to this group the images of the books are not shown (not to overwhelm the visitor).*

In this case, the device context of the user is needed for fulfilling the specified requirement.

6. *A "4\*3 books" offer is shown to the customers. We consider the user is a customer if he has the requirement "Buy Books".*

We need to check whether the user has the requirement "Buy Books". For this purpose we need to store the books bought by the user because we consider the user has this requirement when he has bought one book in the Website.

In Figure 5-1 the Domain and User Model of the system are shown.

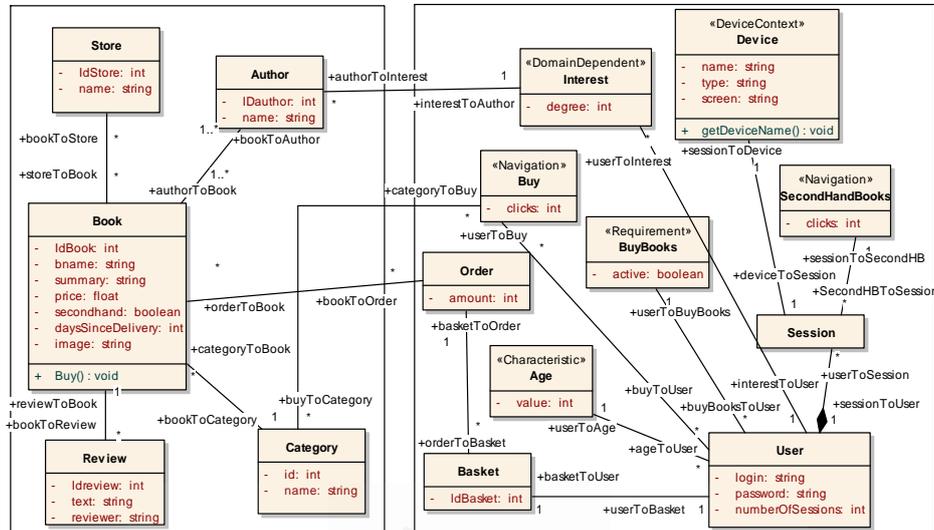


Figure 5-1: Domain and User Model for the (extended) Online Library Case Study

In the updatable data space defined by UM the information describing the user's interests on the authors is stored as *DomainDependent* information, according to the 1<sup>st</sup> defined personalization requirement.

In the UM we also store the books bought on the different categories by the user in the *Buy* class. This class represents a *Navigation event* triggered by the user behaviour and stores the number of clicks done in the link *Buy* in the attribute *clicks* (we will use this information to fulfil the 2<sup>nd</sup> personalization requirement of the running example). In the same way we store the clicks done in the *SecondHandBooks* link, needed to fulfil the 3<sup>rd</sup> personalization requirement, with the class *SecondHandBooks*. The difference with the *Buy* class is that the *SecondHandBooks* clicks are stored for every user session. Note that to store the number of clicks or any other link we would just have to add a new *Navigation* element to the UM.

To cope with the 4<sup>th</sup> defined requirement the age of the User is needed. It is considered a user characteristic (see Chapter 3 for details) and it is stored in the class *Age*.

The Device Context of the user is stored in the *Device* class to cope with the 5<sup>th</sup> personalization requirement specified in this section.

Finally to cope with the 6<sup>th</sup> personalization requirement, it is stored in the UM if the user has or has not the "BuyBooks" requirement.

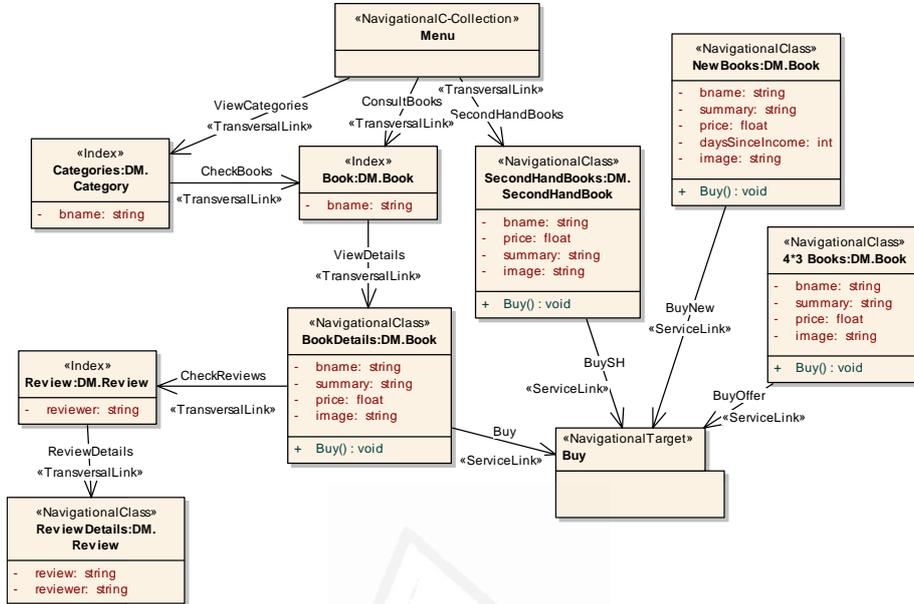


Figure 5-2: Navigation Model for the (extended) Online Library Case Study

Figure 5-2 shows the NM of the (extended) online library. When the user enters the Website he finds a collection of links (i.e. menu) with the links *ConsultBooks*, *ViewCategories* and *SecondHandBooks* as a starting point. If the user navigates through the first link (*ConsultBooks*) he will find an indexed list of all the books (indexed by the book’s name). The user can click on any of the book names to view the details of the chosen book (see in Figure 5-2 the navigational class *BookDetails*). Moreover the user can see reviews done by other users of the different books. When the user clicks on *ViewCategories*, an indexed list of the categories is shown (indexed by the category’s name). When the user clicks on one of the categories he will see the books associated to that category. If the user navigates through the *SecondHandBooks* link he can see all the books used that are on sale.

Finally, there are two more navigational classes “*NewBooks*” and “*4\*3 Books*”. These classes do not have an entry link associated because those links will be created when fulfilling the 2<sup>nd</sup> and 6<sup>th</sup> personalization requirements respectively.

## 5.2 PRML Full Fundamentals

As already explained, PRML Lite is a subset of PRML Full. PRML Full's main purpose is to provide support for the definition of more complex personalization specifications. In this way the methodologies supporting the specification of richer personalization policies (than the ones specified in PRML Lite) can also benefit from the use of PRML. The benefits of using PRML Full are:

- *Easy to learn, to read and to maintain by Web designers:* As already explained, PRML is a Domain Specified Language (DSL) [Van Deursen et al, 2000] which helps to minimize the learning curve for the designer.
- *Possibility of tracking complex browsing behaviour:* PRML Lite already supports personalization on basis of (simple) user's behaviour (i.e. user's click on a link). However, it is important to be able to support the detection of more complex actions of the user. When doing so, some problems have to be faced like how to track the user behaviour.
- *Possibility of adapting the Website for a (dynamic) group of users.*

For being able to use PRML Full, Web methodologies have to support the same minimal requirements specified in the previous chapter (Section 4.2).

PRML Full is based on a MOF [OMG Meta Object Facility] metamodel (see Figure 5-3) which completes the metamodel defined for PRML Lite in previous chapter (Section 4.1) with the full set of constructs of the language. In the PRML Full metamodel all the possible events and actions supported are specified. The different elements of the PRML Full metamodel are explained along this chapter.

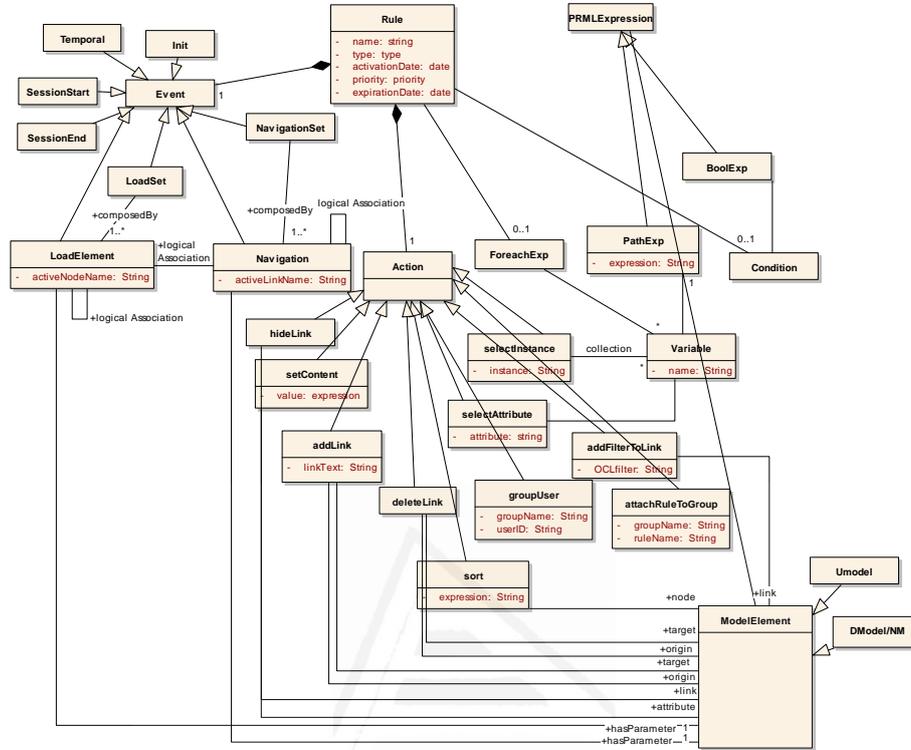


Figure 5-3: PRML Full Metamodel

The basic structure of a rule defined with PRML Full is the following:

```

When event do
    [Foreach expression]
        If condition then
            action1;
            action2; ...
            actionn
        [else
            action1;
            action2; ...
            actionn ]
        endif
    [endForeach]
endWhen
    
```

The difference with a rule defined with PRML Lite is that in the same rule we can define several actions to be performed. This may reduce the number of

rules needed to define a personalization strategy and allows to perform successive actions over the same model element. However the personalization designer has to follow some guidelines in order to write correct rules. These guidelines are explained in Section 5.2.3.

Examples of rules performing several actions are shown in Section 5.2.2. Next we describe PRML Full in terms of the timing of the adaptation and the supported operations over the different model elements.

### 5.2.1 PRML Full: Timing of the Adaptation

Besides the user browsing events supported by PRML Lite, PRML Full supports three new types of events:

- 1 *Temporal events*: Temporal events are independent of the user actions, and can trigger adaptation actions in a certain time interval or at a specific date.
- 2 *System events*: PRML considers a system event called *Init event*, triggered when the application is initialized.
- 3 *Composite events*: To support not so trivial browsing behaviour of the user (i.e. user clicks on a link) more complex (browsing) events are considered, like a sequence of clicks on several links in a specific order, etc. For this purpose PRML Full extends PRML Lite events adding composite events. A composite event is used to combine single events or other composite events. There are three types of composite events:
  - *Sequence of events*: a rule might be triggered when two or more events occur in a particular order. Two subtypes of these events are defined:
    - *NavigationSet event*: It is caused by the activation of a concrete sequence of navigational links (sequence of *Navigation* events).
    - *LoadSet event*: It is associated with the instantiation of a set of nodes (sequence of *loadElement* events).

The sequence of events can be an ordered sequence expressing it with the operator “&” (e.g. “e1 & e2”). It can also be expressed as a non ordered sequence using the operator “|” (e.g. “e1 | e2”).

All the events in the sequence **MUST** be triggered.

The events in the sequence also transfer parameters (the same described in PRML Lite for the single events e.g. “e1(NM.activeConcept par1)“).

- *Logical composition:* (Single or Composite) events can be combined by using *logical operators* **and, or, not**
- *Frequency and Temporal composition:* a rule might be triggered by combination of temporal and non-temporal events, such as “2 minutes after e3” or “3 times e1”. Each event (single or composite) can also specify the number of times it has to be triggered (a link has to be clicked, or a node loaded) and/or the number of seconds the user is browsing the node resulting from the navigation.

The syntax for specifying PRML Full events is described in BNF in the Appendix B of this dissertation. In the next section some examples of the different event types are shown.

#### 5.2.1.1 Events: PRML Full Specification

In this section PRML Full events are described in depth and valid expressions in PRML for the different types of events are shown.

1. *The temporal events* allow triggering the evaluation of the rules without user interaction. The time intervals (i.e. periodicity) considered in PRML are the following:
  - *Always:* the rule condition is continuously checked (i.e. every second). In PRML we express it as: **when Always do**.
  - *Time unit:* the rule condition is evaluated periodically (starting from the day the rule has been triggered) after a specific time unit. The time unit can be defined as a month, week, day or hour. Valid expressions in PRML are:
    - **when Every year do**
    - **when Every month do**
    - **when Every week do**

- **When** Every day **do**
  - **When** Every minute **do**
- *Custom time interval*: the designer can define a concrete temporal expression to state the desired interval of time or specific time to evaluate the rule condition. Some examples of valid custom expressions are the following:

- **When** Every 2 years **do**
- **When** Every 24 hours **do**
- **When** Every 2 days At 13:00 **do**
- **When** Every 23-Feb At 12:00 **do**
- **When** At 23-Feb-2007 **do**
- **When** Every Feb **do**

2. *The system event* (i.e. *Init* event) PRML representation is very simple:  
**When** Init **do**

3. Examples for the PRML *Composite events* are next:

- *Sequence of events: NavigationSet, LoadSet events*

- **When** NavigationSet[NM.ConsultBooks(NM.Book\* b) & NM.ViewDetails(NM.Book b)] **do**
- **When** NavigationSet[NM.ConsultBooks(NM.Book\* b) | NM.ViewDetails(NM.Book b,2,60)] **do**
- **When** LoadSet[NM.Books(NM.Book\* b) & NM.BookDetails(NM.Book b)] **do**
- **When** LoadSet[NM.Books(NM.Book\* b) | NM.BookDetails(NM.Book b,3,60)] **do**

The first example shows a *navigationSet* event. Inside the sequence the operator “&” is used, it implies that all the events in the sequence should occur in the same order specified to trigger the rule.

In the second example the operator “|” is used, so all the events in the sequence should occur (in any order) to trigger the rule. If the triggered event is the second, the *ViewDetails* link has to be activated 2 times, and the user has to be browsing the activated node for at least 60 seconds.

The third and fourth examples are analogous to the first and second ones, but using the *LoadSet* event, what implies that the sub-events are *LoadElement* events (described in Chapter 4, Section 4.2.1).

- *Logical Composition*

- **When** Navigation.SecondHandBooks(NM.Book\* b) **AND** NavigationSet[NM.ConsultBooks(NM.Book\* b)& NM.ViewDetails(NM.Book b)] **do**
- **When** Navigation. SecondHandBooks (NM.Book\* b) **AND** Navigation.ConsultBooks(NM.Book\* b) **AND** Navigation.ViewDetails(NM.Book b) **do**
- **When** Navigation. SecondHandBooks (NM.Book\* b) **AND** NavigationSet[NM.ConsultBooks(NM.Book\* b) | NM.ViewDetails(NM.Book b)] **do**
- **When** LoadElement.ConsultBooks(NM.Book\* b) **OR** Navigation.ViewDetails(NM.Book,1,120) **do**

The first example combines a (single) *navigation* event with a (sequence) *navigationSet* event with the logical operator AND. This means that the two of the events should be triggered to trigger the rule, but the order is not important. In the sequence event, the operator “&” is used, what means that the events inside this complex event should be triggered in the order specified.

The second example combines three (single) *navigation* events. The difference with the previous event is that the sequence event is expressed as single events, so the order of execution is not important. This is equivalent to define a *navigationSet* event with grouping the three navigation events with the operator “|”.

The third example is also equivalent to the previous one. Inside the sequence, the operator “|” is used, what implies that the order of execution is not relevant (but all the events must be triggered).

Finally in the fourth example two single events are combined (*loadElement* and *Navigation* events). Only one of the events should occur to trigger the rule. The *Navigation* event also specifies that the user should click in the link *ViewDetails* once and has to be browsing the node for 120 seconds.

- *Frequency and Temporal Composition*

- **When** Navigation.ConsultBooks(NM.Book\* b,3, 120) **do**
- **When** Navigation.ConsultBooks(NM.Book\* b, 2, 0) **do**
- **When** Navigation.ConsultBooks(NM.Book\* b, 1, 60) **do**

With this type of events we can express things like in the first of the examples shown above: “3 times event e1 and 2 minutes after event e1”. The notation in PRML is **when** e1(NM.activeConcept par1, numberOfTimes, numberOfSeconds) **do**.

In the second of the examples the navigation event is triggered when the user has clicked twice in the *ConsultBooks* link (i.e. 2 times the event resulting from clicking *ConsultBooks*).

In the third example the condition to trigger the rule is that the user has to click on the *ConsultBooks* link once and the user has to be navigating in the node resulting from the navigation for 60 seconds (i.e. 1 minute after the event).

In the next section examples (based on the *online library* case study) to detect more complex browsing behaviour of the user are shown. For this purpose the PRML Full events presented are used.

### 5.2.1.2 Support for Complex Browsing Behaviour in the Case Study

We are going to show the definition of (complex) user behaviour actions by means of examples in the case study. For this purpose, let's consider now the first personalization requirement specified in Section 5.2:

- *Users will see the books sorted by their interest in the books' authors.*

Two things are required here, first is to detect which are the authors in which the user is most interested in, and second would be sorting the book instances.

To detect the interest of the user in the authors we need to track the user behaviour and it is needed to define when we consider that the user has interest on a certain author. This is a subjective issue that can be defined in several ways (considering simple or more complex user behaviour) three alternatives are presented next:

1. *The user has interest on an author when s/he has checked in detail a book of that author.*

In this case, the interest degree on an author is updated when the user consults a book of him. To store/update the value of the *Interest.degree* attribute from the UM the following acquisition rule was defined in the previous chapter (Section 4.2.3.1):

```

When Navigation.ViewDetails (NM.Book book) do
Foreach a, b in (UM.User.userToInterest, book.bookToAuthor)do
  If(b.ID = a.interestToAuthor.ID) then
    setContent(a.degree, a.degree+10)
  endIf
endForeach
endWhen

```

The rule was already explained. Here we only focus on the event part. This is the simplest way of tracking the user behaviour (i.e. user click). When the user clicks on the link *ViewDetails* (see Figure 5-2) the rule is triggered.

2. *The user has interest in an author when s/he has checked at least 3 books of that author and has been navigating at least 1 minute on each of the books pages.*

In this case, the interest degree on an author is going to be updated when the user accesses 3 of the books written by that author and stays at least 1 minute checking each of the three book details. The following acquisition rule is defined to store/update the value of the *Interest.degree* attribute from the UM:

```

When NavigationSet[NM.ViewDetails(NM.Book b1, 1,60) |
NM.ViewDetails(NM.Book b2, 1,60) | NM.ViewDetails(NM.Book b3, 1,60)]do

  Foreach a,b in (UM.User.userToInterest,b1.bookToAuthor) do
    If(b1.bookToAuthor.ID = b2.bookToAuthor.ID and
    b1.bookToAuthor.ID = b3.bookToAuthor.ID and b.ID =
    a.interestToAuthor.ID) then
      setContent(a.degree, a.degree+10)
    endIf
  endForeach
endWhen

```

This rule is triggered when the user activates the link *ViewDetails* three times (for a different book instance) and stays 60 seconds at least in each of the book visited. In the rule condition it is checked that the author of the three book instances visited is the same. It updates the degree of interest in the author of the consulted book using the *setContent* statement, in the same way as the previous rule.

3. *The user has interest on an author when s/he has checked at least 3 books of that author and has been navigating at least 1 minute on each of the books pages or if the user buys one book of that author.*

A new alternative is added for triggering the acquisition rule to update the interest degree on an author: the rule will be also triggered when the user buys the book visited. In this case we need the following acquisition rule:

```

When NavigationSet[NM.ViewDetails(NM.Book b1, 1,60) |
NM.ViewDetails(NM.Book b2, 1,60) | NM.ViewDetails(NM.Book b3, 1,60)] OR
Navigation.BuyBook(NM.Book book) do

  Foreach a,b in (UM.User.userToInterest,b1.bookToAuthor) do

```

```

If(b1.bookToAuthor.ID = b2.bookToAuthor.ID and
b1.bookToAuthor.ID = b3.bookToAuthor.ID and b.ID =
a.interestToAuthor.ID) OR
(book.bookToAuthor.ID=a.interestToAuthor.ID) then
    setContent(a.degree, a.degree+10)
endIf
endForeach

endWhen

```

The event of this rule gets more complex, this rule is triggered when the user activates the link *ViewDetails* three times and navigates over the book details at least 60 seconds or when the user clicks on the *BuyBook* link operation to add the book to the shopping cart. Once it is triggered it updates the degree of interest in the author of the consulted (or bought) book(s) in the same way as previous rules. Note that the condition is extended with an *or* condition for the possible alternatives of triggering the rule.

Once the needed information is stored (user interest degree) we need to sort the book instances based on the user interest stored. This rule is explained in Section 5.2.3.1.

In this section the events supported by PRML Full have been introduced. The conditions that can be defined in PRML Full in the same way as explained in the previous chapter for PRML Lite. In the next section the actions supported in PRML Full are explained.

## 5.2.2 Actions Supported

Besides the functionality supported in PRML Lite the following actions are supported:

- **Actions over nodes (NM):**
  - *Sorting node instances (sort):* In PRML Full it is possible to sort node instances by a certain value to satisfy a personalization requirement.
- **Actions over links (NM):** PRML Full supports adding new links to the NM and deleting them and adding filters on them.
  - *Adding Links (addLink)*
  - *Deleting Links (deleteLink)*
  - *Adding Filters to Links (addFilterToLink)*

- **Actions over users (UM):** PRML Full also allows us to dynamically grouping users by certain criteria to attach personalization rules to them.
  - *Dynamically Grouping users (groupUser)*
  - *Attach personalization rule to a group (attachRuleToGroup)*

Now we explain each of these actions by means of examples on the case study presented.

### 5.2.3.1 Sorting Node Instances (Sort)

To fulfil the first requirement described in Section 5.1 we need to store in the UM the interest of the user on the different authors. An acquisition rule was already defined in previous chapter (Section 4.2.3.1) for this purpose<sup>1</sup>.

```

When Navigation.ViewDetails (NM.Book book) do
  Foreach a, b in (UM.User.userToInterest, book.bookToAuthor)do
    If(b.ID = a.interestToAuthor.ID) then
      setContent(a.degree, a.degree+10)
    endIf
  endForeach
endWhen

```

The personalization rule that sorts the books depending on the interest degree of the user in the book author is:

```

When SessionStart do
  Sort NM.Books orderBy UM.User.userToInterest.degree ASC Where
  UM.User.userToInterest.interestToAuthor.ID= NM.Books.bookToAuthor.ID
endWhen

```

The rule is triggered by a *Session Start event*, when the user enters in the system. The *Sort* action operates over a set of instances. The syntax of this action is very similar to SQL. To properly sort the book instances we need to indicate:

- The node which instances are going to be sorted. Note that the name of the node starts with the prefix NM.

---

<sup>1</sup> Note that we could choose other way of registering the user interest on authors, as shown in Section 5.2.1.2

- *orderBy clause*: this clause allows to sort the node instances by a certain value. In our example we indicate the value by which we are going to sort the book instances: the user interest degree. We can also indicate if the resulting sorted instances are shown in ascending or descending order (ASC or DESC). Additionally we can indicate the maximum number of instances to indicate adding the keyword LIMIT followed by the desired number of instances.
- *where clause*: this optional clause allows to specify a condition which can restrict the results (e.g. regarding the *Recommendations node* if we would want to show only the recommended books in which authors the user has interest degree  $> 100^2$  we would add here a condition checking the interest degree). In this example we just specify the condition needed for properly sorting the book node instances: the correspondence between the interest of the user in the different authors and the books of those authors is checked.

The book instances are shown sorted in the way we have specified with the **PRML Sort action**.

#### 5.2.3.2 Adding Links (*addLink*) and Adding Filters (*addFilterToLink*)

The *addLink* action allows adding a link to the navigation structure of the modelled Web application. It is expressed as: `addLink(linkname, origin, target)`, where it is indicated the text to show in the link, the origin page in which to add the link, and the target page resulting from the navigation for that link (URL). An automatic code is assigned to the link to uniquely identifying it.

If the link we want to add already exists (i.e. same target and same filter or rule attached, if any), it is not added. This is implicitly checked so the designer does not need to specify this condition every time he adds a new rule.

The *addFilterToLink* action allows adding a filter to a link of the navigation structure of the modelled Web application. It is expressed as: `addFilterToLink(filter, link)`, where it is indicated the OCL filter to add to the link and the link to which we want to add the filter. Usually during the navigational design the designer will add the required filters to the NM (as it

---

<sup>2</sup> Note that in this case we would not need the rule specified in section 4.3.1 because the sort action would both filter and sort the instances.

was explained in Chapter 3). However when automatically adding one link (with the *addLink* action) we need to have a way of adding a filter to it.

To illustrate these actions, let's consider the second requirement specified in section 5.1: "If the user has bought 10 or more books of the same category a new link is added to the homepage pointing to the new books on that category."

We need to store in the UM the number of books bought by the user. For this purpose the following acquisition rule:

```
When Navigation.Buy (NM.Book book) do
  Foreach a,b in (UM.User.userToBuy, book.bootToCategory) do
    If (b.ID =a.buyToCategory.id) then
      setContent(a.clicks, a.clicks + 1)
    endIf
  endForeach
endWhen
```

This rule updates the number of books bought on the different categories when the user clicks on the *Buy* link.

The personalization rule has to add a link pointing to a node showing the category books in which the user has bought at least 10 books (in all the sessions).

```
When SessionStart do
  Foreach a in (UM.User.userToBuy)
    If (a.clicks>=10) then
      addLink(a.buyToCategory.name,homepage, NM.NewBooks);
      addFilterToLink(self.daysSinceDelivery<=7 and
        self.bookToCategory.id=a.buyToCategory.id, NM.NewBooks)
    endIf
  endWhen
```

This rule checks the number of books bought by the user and adds a link to the homepage to the new books of the category on which the user has bought 10 or more books.

This rule also adds a filter to the created link, indicating that the attribute *daysSinceDelivery* should be less or equal than 7 and we also have to check that the category of the books to be shown is the same as the one in which the user has bought more than or exactly 10 books. The context of the OCL expression is the target of the link *NewBooks*.

The link is added with the **PRML addLink action** and it is added a filter to this new created link with the **addFilterToLink action**. Note that the destination of the link should exist in the navigation structure.

### 5.2.3.3 Deleting Links(*deleteLink*)

Links can be deleted from the navigation structure of the Website. In PRML it is expressed as: `deleteLink(origin, target)`, in this case it is needed to specify the page where we want to delete the link from (origin) and the target of the link navigation to properly identify the link to be deleted.

To illustrate this action, we consider the third requirement specified in Section 5.1: "The link *SecondHandBooks* is deleted when the user has not consulted it during 10 sessions". For this purpose we need to store in the UM the number of clicks in the *SecondHandBooks* link. The needed acquisition rule is next:

```
When Navigation.SecondHandBooks(NM.Book* books) do
  setContent(UM.User.userToSession.sessionToSecondHandBooks.clicks
    , UM.User.userToSession.sessionToSecondHandBooks.clicks + 1)
endWhen
```

The personalization rule has to delete the *SecondHandBooks* link if we detect that the user is not interested in this section. We consider the user is not interested in the second hand books if he does not click on this link (for being interested we consider he should click at least more than once) in 10 visits to the Website.

```
When SessionStart do
  If (UM.User.numberOfSessions >=10 and
    UM.User.userToSession.sessionToSecondHandBooks.clicks<=1) then
    deleteLink(homepage, NM.SecondHandBooks)
  endIf
endWhen
```

The designer is the responsible of checking possible inconsistencies when deleting a link, being sure that the target destination is accessed by other link if needed.

### 5.2.3.4 Grouping Users

User profiling (i.e. classifying the users into user profiles depending on gathered information about the audience of the Website and adapt the Website considering those groups) is an important issue in adaptive Web applications, due for example to the heterogeneous devices used to access the World Wide Web and the large and heterogeneous audience of Websites. In this context, defining personalization strategies requires proper user grouping that can be either explicit (the user chooses the group he belongs to clicking in the correspondent role shown in the Website, i.e. information for students, information for researchers...) or can be implicitly derived by the system (i.e.

by means of monitoring user browsing behaviour). The explicit classification can be inconvenient for several reasons. One of these reasons is the possible lack of information the user has about these groups (prior to navigation) what can make him being not aware about his role in the Website (e.g. in the context of a university Website a Phd student may not know if his role is *student* or *researcher* as he does not know the information he can find in the subsites for each of the user roles). Furthermore, the classification of the user (his user group) in the Website can change in the course of time (e.g. a customer of a virtual shop can become a privileged customer). Moreover, the user can belong to several user groups at the same time. Hence, it is useful to have a method for implicit (i.e. derived by the system from the user's browsing behaviour) classification of users.

PRML Full allows specifying dynamic user groups based on the user profile. For this purpose a new type of rules is added (called *profile-group rules*). The aim of user grouping is to partition the total user base in a (small) number of groups, for which it is believed the members have some similar needs and goals, or similar motivation to use the site. By mean of these profile-group rules users can be classified into user groups<sup>3</sup>, based on the values of the information defined in their user profiles. A user profile can be defined as data representing the significant features of the user. For defining these groups PRML considers the user information which is independent of the domain. Informal examples of such groups include “*adults*”, “*users with expert domain knowledge*”, etc.

When a user enters the Website the user is attached to user group(s). Obviously, to be able to attach a user to a user group, some information about the user is needed. The required information is gathered by means of acquisition rules. For each defined group, a personalization strategy is specified by attaching personalization rules to it. Once a user is attached to a group(s) only the personalization rules attached to that (those) group(s) are considered (as far as the current user is concerned). To attach a user to a profile group in PRML we write: `groupUser(UM.User,groupname)`. To attach a personalization rule to a profile we write `attachRuleToGroup(rulename,groupname)`.

As explained in Chapter 3 PRML classifies into four adaptivity dimensions the user information which is domain independent: the user characteristics (e.g. user's age, user's hobbies, user's vision level...), the user context which includes different types of context like device context (e.g. PDA, PC, WAP), time context (i.e. date and local time of the connection), network context (e.g. latency, speed, bandwidth and location context (i.e. ubiquity of the user), the

---

<sup>3</sup> A group possibly consists of only one user, and a user can belong to one or more groups.

third dimension are the user requirements (specifying the needs and goals of the user) and the fourth is the user browsing behaviour. Next we discuss how profile-group rules for automated classification of users (for each PRML adaptivity dimension) can be defined.

○ *User Grouping Based on User Characteristics*

In the simplest case, the designer can create a user group for each possible value of a characteristic he wants adaptation support for. E.g. we can define a user group based on the characteristic ‘age’ by using the condition ‘age < 18’ (see below for the exact definition). However, this method may result in many different groups. Ideally, we cluster as many related characteristics as possible to decrease the number of groups. As an example, for most Websites one can imagine that young visitors (age < 12) will not have a good knowledge of domain jargon (domainExperience = ‘low’). Therefore it can be useful to cluster the two characteristics into one profile-group rule.

We now show an example of a profile-group rule that allows to create a group based on a user characteristic. As already indicated, the personalization strategy for the defined group is specified by attaching personalization rules to the group. As an example, we consider the 4<sup>th</sup> requirement specified in Section 5.1: “Non Adults cannot see books of the category *Terror*”. First we group the users with a profile-group rule:

```
When SessionStart do
  If UM.User.userToAge.value <18 then
    groupUser(UM.User , "Minors")
  endIf
endWhen
```

Note that the event that triggers this rule is the SessionStart event: when user enters the system the profile rule will be triggered: if the user fulfils the condition, he is added to the user group “Minors”.

Now we need to define the personalization policy for this group of users, by attaching a personalization rule to it. In this case, we need two personalization rules, one for not showing the books of the category “Terror” when the user clicks on *ConsultBooks* and another one for not showing the category “Terror” when the user clicks on the link *ViewCategories* to check all the possible categories.

```
Rule: RestrictBooks
When Navigation.ConsultBooks (NM.Book* books) do
Foreach b in (books)
  If not (b.bookToCategory.name="Terror") then
    books.SelectInstance(b)
```

```

        endIf
    endforeach
endWhen

Rule: RestrictCategories
When Navigation.ViewCategories (NM.Category* cat) do
Foreach c in (cat)
    If not(c.name="Terror") then
        cat.SelectInstance(c)
    endIf
endforeach
endWhen

```

Now what is left is to say that these personalization rules are only applied to the previously created *Minors* profile group. This is done in the init event, which is launched in the initialization of the Website. Upon this init event, user profiles are defined and personalization rules are associated to them. In this way we specify a personalization strategy for the defined user profiles. In this case it would be as follows:

```

When init do
    attachRuleToGroup("RestrictBooks", "Minors");
    attachRuleToGroup("RestrictCategories", "Minors")
endWhen

```

#### o *User Context*

User Contexts can be represented in a very similar way to user characteristics: they represent the characteristics of a session. For creating user groups based on context, we use a similar approach as for characteristics: start with one group for every distinctive context (with value), and merge existing profile groups if the system determines similarities between different contexts.

Let's consider now the 5<sup>th</sup> requirement described in Section 5.1. We need to define a group of users to classify them if they are using a small screen device (we consider mobile phones, PDAs or MP3 players). The PRML specification of the profile-group rule to define a group of users using a small screen device looks as follows:

```

When SessionStart do
    If (UM.User.userToSession.sessionToDevice.getDeviceName()="PDA" or
        UM.User.userToSession.sessionToDevice.getDeviceName()="MP3" or
        UM.User.userToSession.sessionToDevice.getDeviceName()="Mobile") then
        groupUser(UM.User, "SmallScreenDevice")
    endIf
endWhen

```

When the user enters the site (SessionStart event) this rule calls to a method of the UM called *getDeviceName()* that gathers the device with which the user is browsing the Website. In the condition we check the device name of the users and we group them in a profile group called *SmallScreenDevice* if they are browsing the site either with a PDA, a mobile phone or an MP3 player.

The personalization rule to be attached to the user group *SmallScreenDevice* is next:

```
Rule: PersonalizeDisplay
When LoadElement.BookDetails(NM.Book* books) OR
LoadElement.SecondHandBooks(NM.Book* books) OR
LoadElement.NewBooks(NM.Book* books) OR LoadElement.4*3Books(NM.Book*
books) do
    not(books.Attributes.selectAttribute(image))
endWhen
```

The rule is triggered when the *BookDetails*, *SecondHandBooks*, *NewBooks* or *LoadElement* navigational node is activated. The parameter of the four single events has the same name, because we are performing the same action for the target node instances independently of the event triggered.

This rule does not show the books image when the user is browsing the Website with a small screen device, not to overwhelm the visitor. In PRML all the actions have their opposite action defined and it is specified by negating them with a *not* operator. Now we attach this personalization rule to the previously created *SmallScreenDevice* profile group. This is done in the *init* event, launched in the initialization of the Website. We update the initialization rule as follows (supposing that the name of personalization rule above described is *PersonalizeDisplay*):

```
When init do
    attachRuleToGroup("PersonalizeDisplay", "SmallScreenDevice");
    attachRuleToGroup("RestrictBooks", "Minors");
    attachRuleToGroup("RestrictCategories", "Minors")
endWhen
```

#### o *User Grouping Based on User Requirements*

As in audience driven design [De Troyer, 2005], we aim to cluster users who have the same requirements, since these users will (in principle) be searching for the same information / functionality. User groups based on user requirements will coincide with the so called audience classes in audience driven Web design [Casteleyn and De Troyer, 2001].

Next an example of a profile-group rule that allows to group users based on user requirements is shown. For this, consider the 6<sup>th</sup> requirement specified in

section 5.1. The following profile-group rule defines the *Customers*. The *Customers* have one requirement: buy books.

```
When SessionStart do
  If (UM.User.userToBuyBook.active="true") then
    groupUser(UM.User, "Customers")
  endIf
endForeach
endWhen
```

We consider that the user has the requirement “*buyBooks*” once he buys a book in our Website (i.e. once the visitors buy a book they are considered customers). So in order to update the UM with this information the following rule is defined:

```
When Navigation.Buy(NM.Book b) do
  Foreach a in (UM.User.userToSession.sessionToBuy)
  If (a.clicks>0) then
    setContent(UM.User.userToBuyBook.active, "true")
  endIf
endWhen
```

The event that triggers this rule is again the Start event: when user enters the system the profile rule will be triggered. If the user has been identified to possess the requirements specified in the condition part of the rule, the current user will be added to the profile-group.

The personalization rule to be attached to the profile group *Customers* is next:

```
Rule: Show4X3Offer
When SessionStart do
  addLink("Offer:4*3 Books",homepage,NM.4X3Books)
endWhen
```

This rule adds a new link when the user enters the Website, showing the books with the offer 4X3. This personalization rule is only applied to the previously created *Customers* profile group. The same as the other examples, we attach the rule to the user group in the *init* event, which is launched in the initialization of the Website. We add the following action to the initialization rule described in previous sections (supposing that the name of personalization rule above described is *Show4x3Offer*):

```
When init do
  attachRuleToGroup("Show4X3Offer", "Customers");
  attachRuleToGroup("PersonalizeDisplay", "SmallScreenDevice");
  attachRuleToGroup("RestrictBooks", "Minors");
  attachRuleToGroup("RestrictCategories", "Minors")
```

endWhen

As the reader sure has noticed, we haven't specified still how to group users by their browsing behaviour. Instead of grouping the users by their browsing behaviour PRML Full supports the definition and detection (at runtime) of user behaviour patterns. The consequence is that the complexity of the personalization strategies to define (based on user browsing behaviour) increases. The detection of user navigational patterns (defined for a very common user behaviour) at runtime implies some problems like how to track the user behaviour and how to recognize certain behaviour pattern. How to track (complex) user behaviour and recognizing patterns is explained in next section.

#### 5.2.3.5 Support for Behaviour Patterns

Different behavioural patterns are possible in the browsing behaviour of the visitors. In order to better accommodate the users, we can analyze their behavioural patterns, and adapt the site accordingly if we recognize a certain pattern (over a significant amount of time). The definition of such patterns makes easier the tracking of complex user browsing behaviour.

To support behaviour patterns definition and recognition in PRML we add a new type of rules, called *behavioural rules*. These rules track the user browsing behaviour and detect (at runtime) navigational patterns (defined also with *behavioural rules*). Moreover when a pattern is detected the proper action is performed. The main difference with the rest of PRML rules (i.e. *acquisition* and *personalization rules*) is the parameters passed in the events. In this kind of rules we pass (as a parameter) the navigational path that the user is following. We need to define (for each defined pattern) what we consider a navigational path. We show next an example for a browsing behaviour patterns defined in [Casteleyn et al, 2004].

#### **Direct Path pattern:**

*Intent:* provide direct navigation access to information relevant for the particular user, instead of forcing the user to follow each time the same set of navigation tracks.

*Solution:* adapt the navigation access point of the user if a direct path from that access point to a certain (other) node is detected in a certain percentage of session (for example, 80%) or in a significant number of times. In that case, a link to the relevant information is added.

*Consequence:* reduces amount of clicks to relevant information (for the particular user)

When a user enters in the system we start tracking his/her behaviour, keeping the navigational paths that s/he follows. We (only) consider a (finished) navigational path (for this browsing pattern) a path that starts in the page where the user entered the application and finishes in the page in which s/he stays at least 2 minutes. We can see a representation of the navigational path for this pattern in the statechart of Figure 5-4.

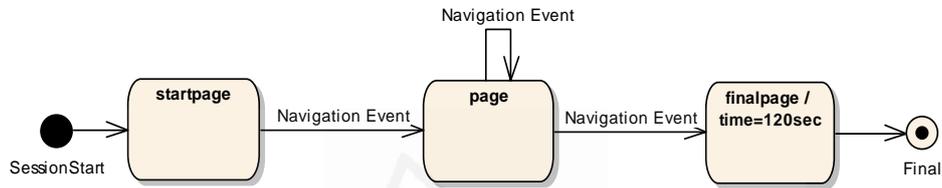


Figure 5-4: Navigational Path for the *Direct Path* pattern

The navigational paths are stored in the UM, each of them having an identifier. A behavioural pattern can be defined as a certain sequence of navigation actions. Concretely the Navigational path for the *Direct Path* pattern is built from the triggering of Navigation events by the user. We should add a class to store the particularities of the *Direct Path* pattern to the UM as follows:

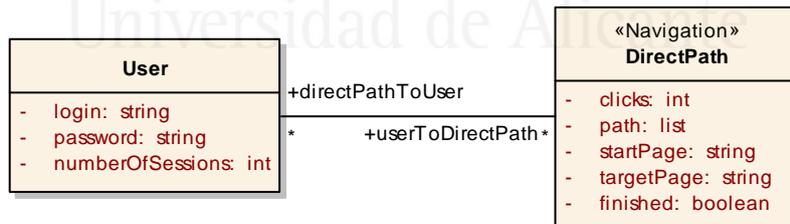


Figure 5-5: UM class to be added for the *Direct Path* pattern detection

In the class added (i.e. *DirectPath*) the following information about the navigational path is stored:

- *clicks*: the clicks done in the detected path are stored. We consider that the *Direct Path* pattern is applied when the user has more than 100 clicks on the detected navigational path.

- *path*: the navigational path of the user is stored in the path attribute.
- *startPage*: the starting page of the navigational path is stored.
- *targetPage*: the target page of the navigational path detected is stored.
- *finished*: if the navigational path has been finished, this boolean attribute is set to true.

We define the following set of PRML rules to detect the direct path pattern:

### PATH CONSTRUCTION

Three behavioural rules are defined to construct the navigational path for the *Direct Path pattern*.

```

Rule:PathInitialization
When SessionStart do
setContent(UM.User.userToDirectPath.startPage, startpage)
endWhen

```

This rule is triggered when the user enters the Website and it initializes the navigational path of the user with the starting page (i.e. the page in which the user starts the session on the Website, represented by the keyword *startpage*).

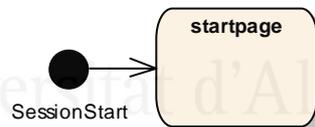


Figure 5-6: Path Initialization

Next we need to build this navigational path meanwhile the user is browsing the Web application. For this purpose we have the following *behavioural rule*:

```

Rule:PathConstruction
When Navigation.Link(NM.CurrentPath path) do
Foreach a in (UM.User.userToDirectPath) do
If (a.path=path) then
    setContent(a.path, a.path+Link)
endif
endWhen

```

This behavioural rule is triggered when the user activates any link (represented by the *Link* id). As a parameter (as already stated) we have the current path of the user in the Website. The path stored in the UM is updated, appending the new browsed link to the navigational path (stored as a list).

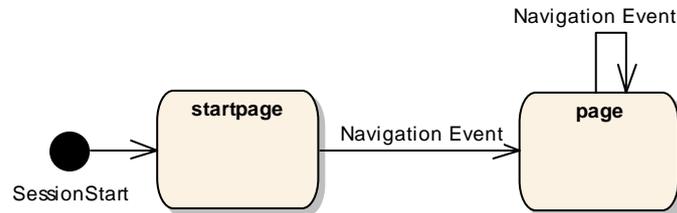


Figure 5-7: Path Construction

What is left now is to detect when the path has been “finished”. As aforementioned we consider a navigational path being finished when the user browses a page for at least 2 minutes. We express this with the following *behavioural rule*:

```

Rule:PathCompletion
When Navigation.Link(NM.CurrentPath path ,1,120) do
Foreach a in (UM.User.userToDirectPath) do
  If (a.path=path) then
    setContent(a.path, a.path+Link);
    setContent(a.finished, true);
    setContent(a.targetPage, Link.target)
  endIf
endForeach
endWhen
    
```

This behavioural rule is also triggered by any link activation (represented by the *Link* id). In this case we add to the path (list) variable of the UM the last link visited. Moreover we identify the visited path as a finished path and we store the target page of the navigational path.

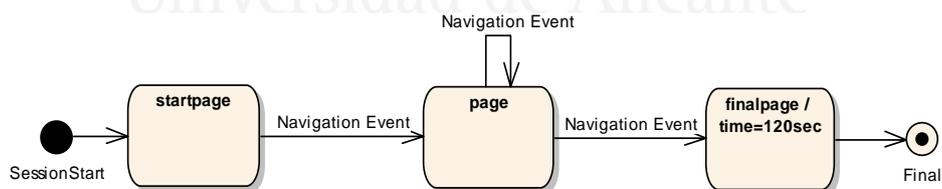


Figure 5-8: Path Completion

### PATH BROWSING DETECTION

Once a browsing path is detected, we need now a rule to update the clicks on the finished path when the user goes through it:

```

Rule:PathDetection
    
```

```

When Navigation.Link(NM.CurrentPath path) do
Foreach a in (UM.User.userToDirectPath) do
  If (a.path=path and a.finished=true) then
    setContent(a.clicks, a.clicks+1)
  endif
endForeach
endWhen

```

This rule checks if the current visited path has been already detected as a finished one (checking the last element of the list in which the path is stored) and if so the number of clicks on is increased.

#### DIRECT PATH PATTERN DETECTION

What is left now is to check if the direct path pattern is detected and perform the proper action. We consider that if a user has more than 100 clicks on a navigational path the direct path pattern is detected, and what we do is to add in the home page a shortcut to that path.

```

Rule:DirectPathPattern-Detection
When SessionStart do
Foreach a in (UM.User.userToDirectPath) do
  If (a.clicks>100) then
    addLink(a.targetPage, homepage, a.targetPage)
  endif
endForeach
endWhen

```

Note that we add the link in the home page containing as text the name of the target page of the navigational path.

In the same way as we have done with the *Direct Path pattern* different browsing behaviour patterns can be defined and recognized using behavioural rules.

### 5.2.3 Designer Guidelines

As explained in Section 5.2, in a PRML Full rule we can define several actions to be performed. This may reduce the number of rules needed to define a personalization strategy and allows to perform successive actions over the same model element. However the personalization designer has to follow some guidelines in order to write correct rules:

- *Acquisition rules* can only contain actions updating the data on the DM/UM (i.e. *setContent* action).

Multiple *acquisition rules* sharing the same event should be avoided. The reason is that when unsynchronized data updates take place can lead to possible data inconsistency.

- In the *Personalization rules* the actions should be defined taking into account that:
  - Filtering attributes (i.e. *selectAttribute*) should be defined before than filtering instances (i.e. *selectInstance*).
  - Sorting node instances (i.e. *sort*) should be defined after filtering attributes or instances over a node.

Note that the actions defined in the same PRML rule should operate over the same model element, otherwise different PRML rules should be defined.

Finally it is important to say that composite events are used in the PRML rules if the personalization designer needs to define an adaptivity action when one or more browsing behaviour events are triggered or after a certain sequence of user browsing events.

### 5.3 Conclusions

In this chapter, the fundamentals of PRML Full have been explained. PRML Full provides the designer a way of specifying more complex personalization actions than PRML Lite (presented in Chapter 4) limiting the reusability. PRML Full actions are needed for specifying more advanced personalization strategies. For this purpose it also allows recognizing complex navigation behaviour of the user, needed for defining efficient personalization strategies. As future work we plan to extend PRML Full to support adaptation actions over the presentation of the Website.

As already explained in Chapter 4, the main purpose of PRML Lite is making possible the reuse of personalization strategies among different approaches. The portability of PRML is explained in chapters 7 and 8.

In the next chapter a prototype CAWE tool for the automatic generation of adaptive Web applications using PRML is described. Once generated, the adaptive Website also contains two modules for managing the personalization which, at runtime, analyze the user browsing events and adapt the Website according to the personalization rule(s) triggered.

## Chapter 6

### Automatic Generation of Adaptive Websites

The previous chapters presented the fundamentals of PRML Lite and PRML Full. This chapter's objective is to present the prototype tool able of generating dynamically adaptive Web applications from A-OOH models. This tool generates code from the A-OOH design models presented in Chapter 3. Moreover it is able of interpreting the PRML rules attached to those models, which define personalization policies for the users.

During the use of the generated Web applications, these will gather and analyze information about the activity of the user in the system and/or his interaction environment. Depending of that information, the presented interface will adapt to the needs and interests of the user.

The ad-hoc development of Web-based systems lacks a systematic approach and quality control. Web Engineering, an emerging new discipline, advocates a process and a systematic approach to development of high quality Web-based systems. So far, few of these approaches provide an underlying CAWE<sup>1</sup> tool for Web Engineering and even less provide a tool to support personalization modeling (see Section 6.1 for an overview). The lack of such tools causes the personalization to be implemented in an ad-hoc manner. Moreover, the different (adaptive) methodologies are not properly tested yet.

In this chapter, we present the fundamentals of the AWAC (Adaptive Web Applications Creator) prototype tool. As aforementioned this is a CAWE tool which automatically generates adaptive Web applications based on the A-OOH methodology presented in Chapter 3. The input of the AWAC tool is the set of A-OOH design models needed to model the adaptive Website to

---

<sup>1</sup> Computed Aided Web Engineering

generate. The output is the generated Website with its database and includes a Web engine and a personalization module which allow the adaptivity of the final Web pages at runtime. The chapter is structured as follows. In the following section, a study of the existing methodologies with a CAWE tool supporting adaptivity is presented. In Section 6.2 the architecture of an application generated with AWAC is described. Along the Sections 6.3, 6.4, and 6.5 the different steps for creating and running a Website using the AWAC tool are explained. The running example that is used along the dissertation (online library) is modified to describe the tool support. Section 6.6 will describe how the *PRML manager*, a module of AWAC for updating the personalization strategies of a Web application at runtime explained in Section 6.2, is used to update the personalization rules when the application is running. Section 6.7 presents an experiment done with AWAC and the results are discussed. Finally, Section 6.8 sketches the conclusions of the chapter.

## 6.1 Related Work

As aforementioned, few (adaptive) Web modeling approaches provide an underlying CAWE tool to give support to their methodologies. We can mention the Hera Presentation Generator (HPG) [Fransiscar et al, 2006], which is the integrated development environment that supports the Hera methodology developed at the Technical University of Eindhoven (The Netherlands). There are two versions of HPG: HPG-XSLT and HPG-Java. Compared with HPG-XSLT, HPG-Java extends the functionality of a generated Website with user interaction support (form-based). Moreover, instead of generating the full Web presentation like HPG-XSLT does, HPG-Java generates one-page-at-a-time in order to better support the dynamic Web applications. The designer can define adaptation by means of the inclusion of appearance conditions over the elements of the Hera design models. These conditions are expressed in SerQL [OpenRDF] language and use data from the user/platform profile or conceptual model. A drawback of this approach is the difficult maintenance when the personalization policies change because the conditions are integrated in the models.

Another tool for generating adaptive hypermedia applications on the WWW is AHA! [De Bra et al, 2003] (Adaptive Hypermedia Architecture), based on the

AHAM model. It also has been developed at the Eindhoven University of Technology (The Netherlands). It is able to perform adaptation that is based on the user's browsing actions. AHA! provides adaptive content by conditionally including page fragments, and adaptive navigation support by annotating (actually coloring) links. The updates to attributes of concepts in the user model are done through event-condition-action rules. Every rule is associated with an attribute of a concept, and is "triggered" whenever the value of that attribute changes. Every page has an access attribute which is (virtually) "changed" whenever the end-user visits that page. This triggers the rules associated with this attribute. The AHA! tool claims to be general purpose but has mainly been used to develop on-line courses.

We can also mention WebRatio [WebRatio] developed to support the WebML methodology at the Politecnico di Milano (Italy). This tool still does not support dynamic personalization features, but only adaptability (with respect to user profile/preferences and device characteristics). However some running prototypes have been developed, described in [Ceri et al, 2007; Daniel et al, 2007]. To validate WSDM, at the University of Brussels (Belgium), a prototype tool was created for the support of the methodology [Casteleyn, 2005]. It does not support personalization, but adaptivity for all the users. ArgoUWE [Knapp et al, 2003] is the tool developed to support the UWE approach at the Ludwig Maximilians University of Munich (Germany). UWE supports personalization however it is not yet incorporated on the ArgoUWE tool.

## 6.2 AWAC: Adaptive Web Applications Creator

The AWAC tool main purpose is automatically generating an adaptive Web application from the A-OOH models. The AWAC tool takes as *input* the A-OOH design models: the *Domain Model*<sup>2</sup>(DM), in which the structure of the domain data is defined, the *Navigation Model* (NM) which defines the structure and behaviour of the navigation view over the domain data, and finally the *Presentation Model* (PM) defines the layout of generated hypermedia presentation. To be able to model adaptation/personalization at

---

<sup>2</sup> Sometimes called Conceptual Model

design time two new models are added (to the set of models): Firstly, the *User Model* (UM), in which the structure of information needed for personalization is described. Typically, the information captures beliefs and knowledge the system has about the user and it is a foundation for personalization actions described in the Personalization Model. Secondly, the *Personalization Model* (PeM), in which personalization policies are specified. Next to the personalization of the content, navigation structure and presentation, the personalization model also defines updates of the user information specified in the User Model.

These models are represented by means of XML elements (in XMI [XML Metadata Interchange] format). The reason for choosing an XMI representation of the models is that this format can be easily generated from UML models<sup>3</sup>. To read and process the A-OOH models for the generation of the final Web pages we have used the .NET technology. This technology provides us with the DOM class (XML Document Object Model), with which we can represent in memory the XML documents.

The *output* of the AWAC tool consists of:

- **The generated adaptive Website (Web pages):** the current version of AWAC generates ASP.net Web pages.
- **Modules for managing the personalization:** these modules are explained in the next section.
- **Application database:** The A-OOH models initially represented in XMI models are mapped into an object oriented database. Depending on the personalization actions performed every user has a different set of A-OOH model instances. This database also contains the user information related to the domain. The idea of using a relational database was rejected due to the complexity transitioning from object-oriented thinking to relational persistence. In this way the database can automatically be generated from the set of A-OOH models. The database technology we chose is db4o [Db4o], the database for objects of open code most popular, native to Java and .NET. Db4o eliminates the OO-versus-performance

---

<sup>3</sup> Most UML tools allow this transformation

tradeoff: it allows you to store even the most complex object structures with ease, while achieving highest level of performance.

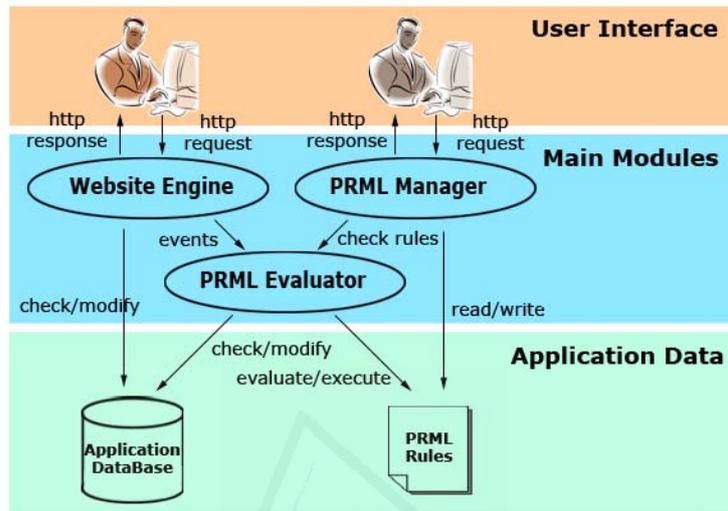


Fig. 6-1. Generated AWAC Application Architecture

### 6.2.1 Generated AWAC Application Architecture

The generated Web Application has a three layered architecture as shown in fig. 6-1.

- The first layer is the *user interface*, the user can generate http requests and receive http responses.
- The second layer contains the main modules of the Web Application for managing the personalization (i.e. *Website Engine*, *PRML Evaluator* and *PRML Manager*, see fig. 6-2). The *Website Engine* interacts with the user, gathering the requests and giving back the response. As already mentioned, the A-OOH models are modified for each particular user (i.e. each user will have a different set of models depending on the adaptation actions performed on them). The *Website engine* loads the models (from the Application Database) of the particular user when s/he starts a new session. These models are modified along the different sessions depending on the adaptation actions performed. This implies that each user will see a different adaptive view of the Website every session s/he

browses it. The *Website engine* captures the events that the user performs with his/her browsing actions and sends them to the *PRML Evaluator* module. This module is responsible for evaluating and performing the personalization rules attached to the events. When a rule is triggered, to evaluate the rule conditions and perform the proper actions we have implemented a .NET component using the ANTLR Parser generator [ANTLR]. From the PRML grammar we can generate the syntactic trees which help us evaluate the rule conditions and perform them if necessary. Finally, to execute the actions of the rules, we have implemented into C# the different action types that we can find in PRML. The adaptive actions are only performed once during a session not to overwhelm the user. This means that the adaptive information will be the same during the present session. This is the option by default; however, it is also possible that the designer decides when an adaptation should take place to fulfill each personalization requirement. How to specify this is explained at the end of Section 6.2.3. The *PRML Manager* module will be explained in next section.

- The third layer contains the Application database and a text file composed of the set of rules defining personalization policies on the Website. This set of rules is defined using the PRML rule. Next the implementation of the PRML actions supported by AWAC is explained.

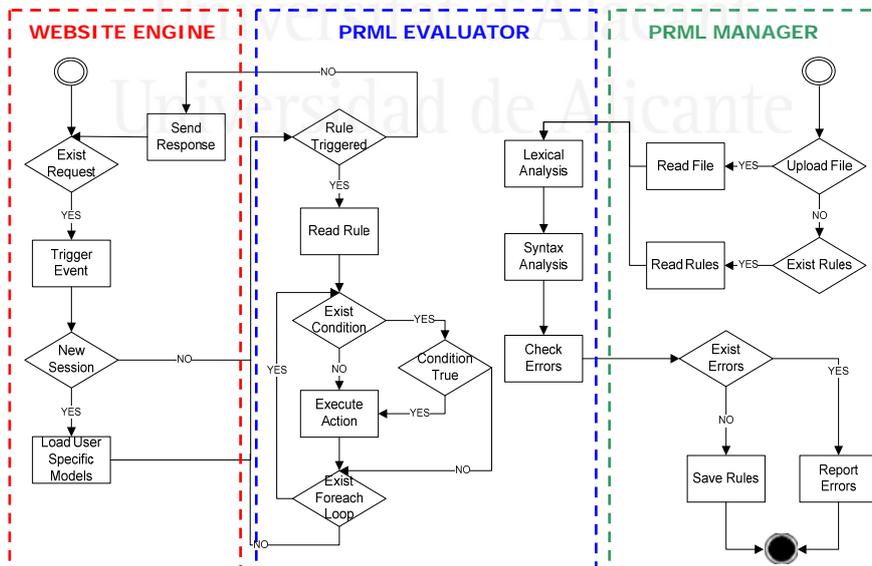


Fig. 6-2 Main Modules Actions

### 6.2.2 PRML Manager: Personalization Management at Runtime

In PRML, personalization is defined at design time and performed at runtime. However, a problem arises for the designer when s/he wants to update a rule at runtime: is it necessary to re-generate the whole Web application?. It would be necessary if the rules were embedded in the design models. When personalization is tightly intertwined with the rest of the application and the underlying technology some problems arise such as the difficulty of personalization maintenance and the difficult evolution of the adaptive Web site.

In most of the website modeling methods which allow developing personalized websites, personalization specifications are embedded in their design models causing the problems described above. In Web applications generated with AWAC personalization is defined at design time but it can also be modified at runtime. The management of personalization at runtime requires that the rules defined to satisfy the adaptivity requirements are totally independent of the design models. This independence provides several advantages such as an easier maintenance and evolution of the Web site.

PRML rules are defined in a separate file and can be modified at runtime. For this purpose we have created a module in AWAC called *PRML manager* which allows us to update rules at runtime. The PRML manager is a Web page which can be accessed by designers to update the rules of the website. The screenshot of its main page is shown below.



Fig. 6-3. PRML Manager: Sign-in

The main screen is where the personalization designer signs in. After that, s/he can see the Web page shown in Figure 6-4. On this Web page, the designer can load a new PRML file to be used with the Web application that is being run. S/He can also update the existing set of rules. For this purpose, the designer will directly modify the text of the PRML file and once finished s/he will press the “Save rules” button. If a lexical or syntactic error is found the file will not be saved and errors will be shown in the STATUS section (see Figure 6-4). The *PRML evaluator* module checks for lexical and syntactic errors in the PRML file:

- It checks that the tokens used in the rule are those specified in the PRML grammar. Otherwise, it indicates that a lexical error has happened indicating the row and column where the unexpected token has been found. For example: `PRMLLogger: rules:4:29: unexpected char: '@'`
- It also checks that the rule is correctly built, so the tokens are used taking into account the rules of the grammar. Otherwise, it indicates where the syntactic error has been found (row and column), for example: `PRMLLogger: rules:2:37: expecting "do", found 'When'`

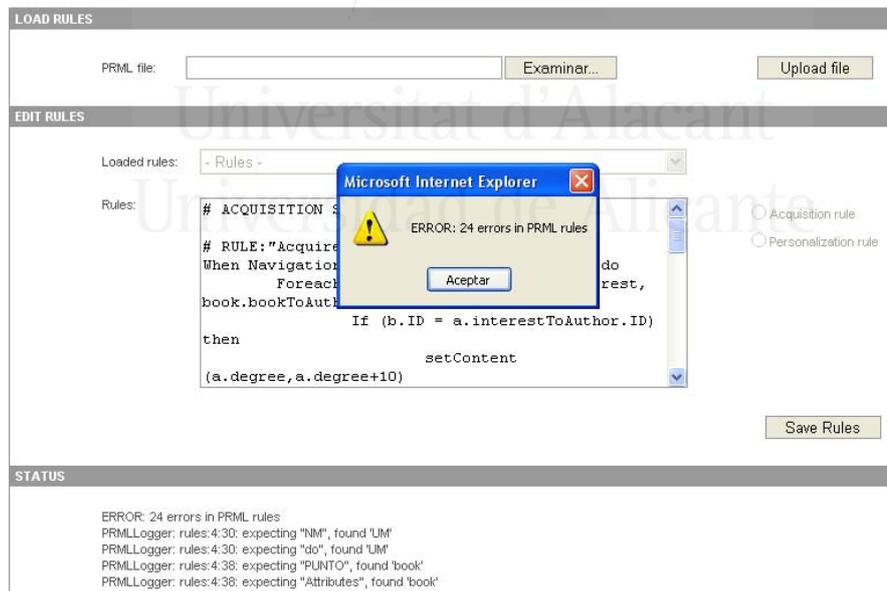


Fig. 6-4. PRML Manager

The designer can modify the rules performing the following actions:

- Adding a new rule

The designer can add a new rule to the file. This new rule should refer to modeling elements of the models already defined in the Web application but not to new ones; otherwise we should update also the models and regenerate the Web application.

- Deleting an existing rule

Besides, the designer can delete an existing rule; however s/he has to be sure that there are not conflicts produced by this action. As a future work we plan to automatically check the possible inconsistency with the tool.

- Updating a rule

Another action the designer can perform is that of updating a rule of the PRML file:

- The designer can update the rule so that it is triggered by a different event.
- The condition can also be updated if the designer decides that the criteria for personalization should change.
- The action can also be modified but taking into account that in this case the structure of the rule can change.

Semantic errors are not checked so the designer should take this into account when updating the rules. Next, the implementation of the PRML actions supported by AWAC will be explained. In Section 6-7 an example of personalization management at runtime will be shown.

### 6.2.3 AWAC: PRML Support

The AWAC Tool does not implement all the adaptation events and actions of PRML. AWAC only supports the detection of simple browsing events (i.e. not a sequence of events). The events supported are: *SessionStart* (triggered with the start of the browsing session by a user) *SessionEnd* (triggered when the browsing session expires after certain inactivity of the user in the system or when the user explicitly finishes the session), *Navigation* (i.e. click on a link)

and *LoadElement* (i.e. loading of a navigational node independently of the link that loads the node<sup>4</sup>). In Table 6-1 the personalization actions supported by PRML and the ones implemented in AWAC are shown. Next the actions supported by AWAC are detailed.

**Table 6-1:** PRML Support in the AWAC tool

<i>Action</i>	<i>PRML</i>	<i>Implemented in AWAC</i>
<b>Updating User Model Content</b>	Yes	Yes
<b>Filtering content (concept instances)</b>	Yes	Yes
<b>Filtering content (concept attributes)</b>	Yes	Yes
<b>Link hiding</b>	Yes	Yes
<b>Sorting content (node instances)</b>	Yes	Yes
<b>Adding / deleting links</b>	Yes	No
<b>Adding filters to links</b>	Yes	No
<b>Dynamically grouping users</b>	Yes	No

The AWAC tool implements the following actions over the different elements of the A-OOH models:

- Actions over attributes (User and Navigation Models):
  - *Updating an attribute value from the User Model (setContent)*: This action allows modifying/setting the value of an attribute (of a concept) of the User Model. To perform a *setContent* action the PRML Evaluator updates the corresponding attribute value in the Application database.
  - *Filtering attributes in the Navigation Model nodes (selectAttribute)*: By means of this action a node can be restricted by hiding or showing some of the attributes of the Domain Model/User Model related concept. This action affects to the visibility of an attribute, each attribute in A-OOH contains a visibility property. The PRML Evaluator updates the visibility of the corresponding attribute in the proper model of the Application database.
- Actions over links (Navigation Model):
  - *Hiding links and their target nodes (hideLink)*: Analogous to filtering data content, PRML also supports filtering links. This action affects to the

---

<sup>4</sup> Note that PRML rules can be attached to nodes or to links of the Navigation Model

visibility of a link so in the same way as attributes, each link and node in A-OOH contains a visibility property. The PRML Evaluator updates the visibility of the corresponding link (and its target node) in the proper model of the Application database.

- Actions over nodes (Navigation Model):
  - *Filtering node instances (selectInstance)*: This action allows to show only the selected instances of a Domain Model/User Model concept for a user depending on the personalization requirements we want to support. The PRML Evaluator module selects the instances to be shown in the actual session from the Application database.
  - *Sorting node instances (sort)*: In PRML node instances can be sorted by a certain value to satisfy a personalization requirement. The PRML Evaluator module selects and sorts the instances to be shown in the actual session from the Application database.

As explained in Section 6.2.1, by default, the adaptive actions are only performed once during a session not to overwhelm the user. This means that the filtered attributes, links, instances and sorted instances will be the same during the present session. The designer can change this default option and decide when an adaptation should take place to fulfill each personalization requirement (i.e. runtime, twice during a session, etc). For this purpose the designer has to modify the first row of the PRML file. In this row, we can find the information about the version of PRML being used and the frequency of performance of the rules during a session. This attribute can take the values from 1 to n, where n means that the personalization is continuously performed. (i.e. when each event is triggered).

```
PRMLversion:Full PersonalizationFrequency: 1
```

We can also specify in every rule, the frequency of execution of that rule during a session. In this way, each rule can have different frequency of performance specifying it in the features of the rule:

```
Rule: HideImageRecommend Type: Personalization
PersonalizationFrequency:n
```

Next, by means of a running example, the steps needed for creating and running a Website using AWAC are described.

## 6.3 Experiment Setup

### 6.3.1 Step 1: Creating the A-OOH Models

To better understand how to generate adaptive Websites with AWAC using A-OOH and PRML, the online library case study used along the dissertation has been modified to be put online. The objective of this experiment is twofold: to evaluate the satisfaction of the users (in terms of personalization performed, fast response...) and the improvement of the personalization techniques applied.

The case study of the experiment (*bibliography system*) is based on the *online library* case study. This example has been adapted to the needs of the students of the University lab of the author of this dissertation. The Web application was generated by the AWAC tool and was put online. In this Web application the users (students) can browse the books recommended in the lab for the different courses taught. They can also add reviews on the different books, and consult the books by topic. The personalization requirements considered were the following:

1. *Users will see recommendations of books in which topics they are most interested in (sorted by interest).*
2. *The user will see topic related links in which topics the user is most interested in.*
3. *If the user does not have enough interest in any topic to get personalized recommendations, the link recommendations will not be shown:* In order to fulfil the 1<sup>st</sup>, 2<sup>nd</sup> and 3<sup>rd</sup> requirements we need to acquire and update the user interest in the different topics.
4. *In the same way as requirement 3, the user will not see the LinksOfInterest link if he does not have enough interest.* To cope with this requirement we also have to store the user interest on the different topics.
5. *If the user is using a mobile device (i.e. mobile or PDA) the images of the books are not shown:* In this case, to fulfil this requirement (and hide/show the books images to the user) the device context of the user should be stored in the UM.

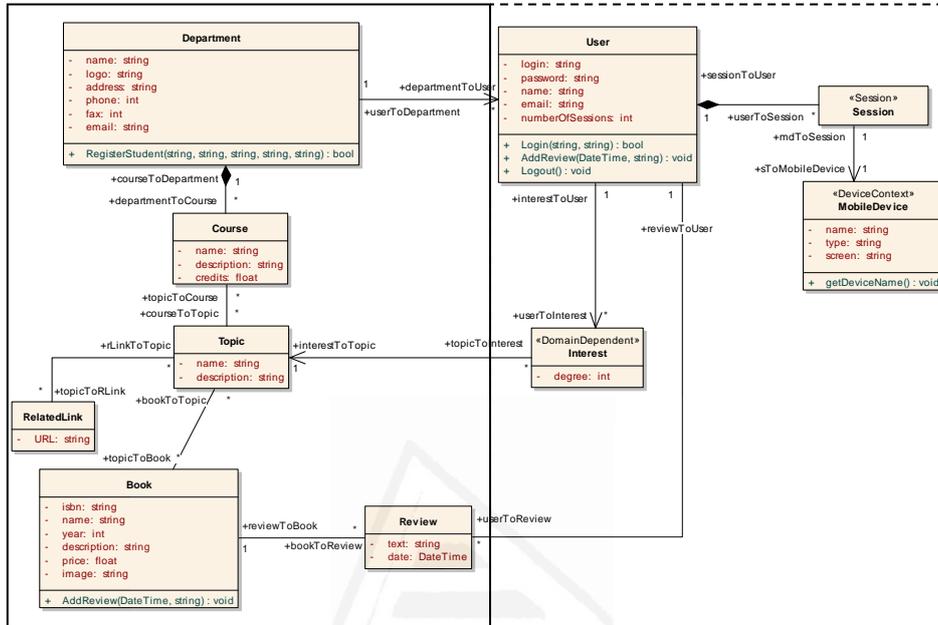


Fig. 6-5. Domain and User Model for the Bibliography System

As aforementioned, the user-related updatable information needed for personalization is stored in the User Model:

- Information describing the user’s interests on the different topics.
- The device context of the user is stored to cope with the 3<sup>rd</sup> requirement specified.

Figures 6-6 and 6-7 show the Navigational Model of the *bibliography* system. Figure 6-7 shows the explosion of the navigational target *System.Private* which contains the private part of the website. When the user enters the Website (login) he finds a collection of links (i.e. menu) with the links *Courses*, *Topics*, *Books*, *Recommended Books*, and *Links of Interest* as a starting point. If the user navigates through the first link (*Courses*) he will find an indexed list of all the courses (indexed by their name). The user can click in any of the course names to view the topics in which that course is classified, and from them the user can consult the books contained in each of the topic. Moreover the user can see reviews done by other users of the

different books. When the user clicks on *Topics*, an indexed list of the topics is shown (indexed by the topic's name). When the user clicks in one of the topics he will see the books associated to it. The user can also directly consult the set of books by clicking in the *Books* link. Moreover he can see *Recommended Books* depending on his interest on the different topics (as specified in the first personalization requirement of the case study). If the user navigates through the *Links of interest* link he can see a list of related links of the topics he is most interested in.

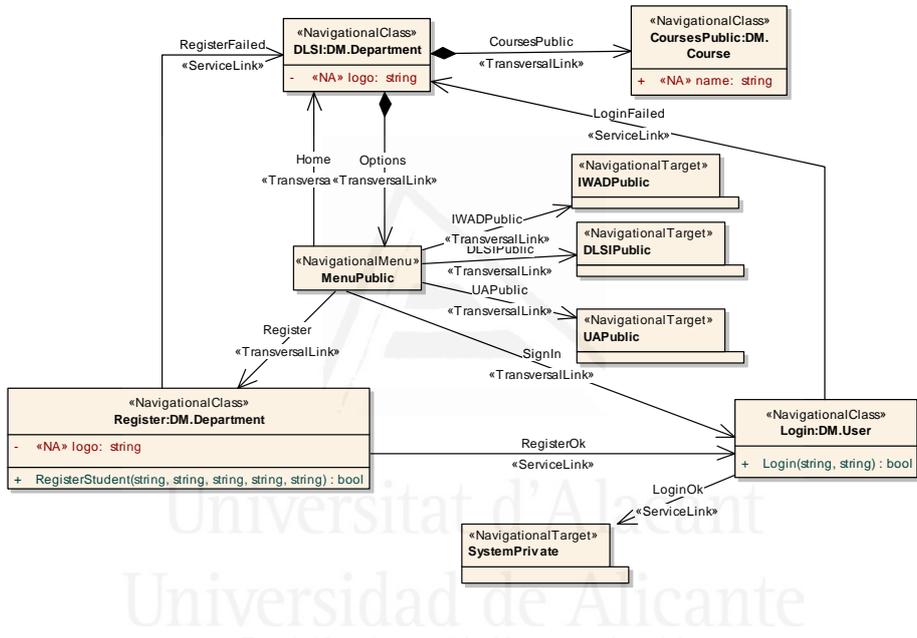


Fig. 6-6.Level zero of the Navigational model

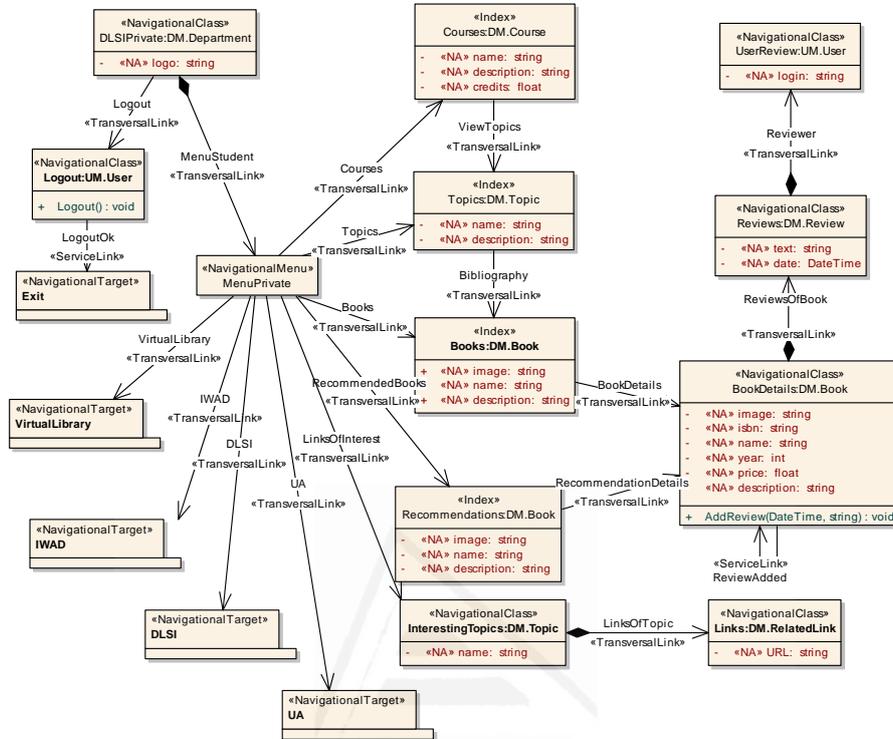


Fig. 6-7. Navigational Model: explosion of the System Private package

Once the NM is specified the Presentation Model has to be defined. In Figure 6-8 the DPD (level 1) for the page that shows recommendations to the user. This page is defined as a set of layouts and cells. The layouts are composed of cells which contain interface components that represent the elements of the Web page. As explained in Chapter 3, the Page Chunk element represents a fragment of an abstract page which has associated a Presentation model where the components shown in this fragment are defined. This fragment can be reused in the different pages that compose the Web application. Thus, we avoid to make several diagrams of common parts to all (or many of) the pages. Inside this package the Presentation Model attached to the Page Chunk is shown. Figure 6-9 shows the Presentation Model attached to the MenuPrivate page chunk. In this case, the Menu is defined as a page chunk.

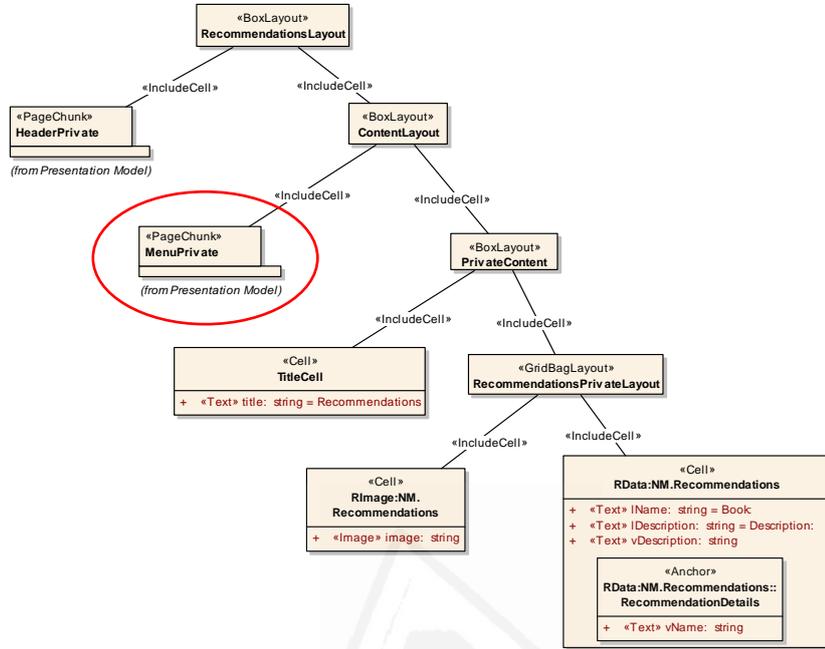


Fig. 6-8. PM for the Recommendations Page

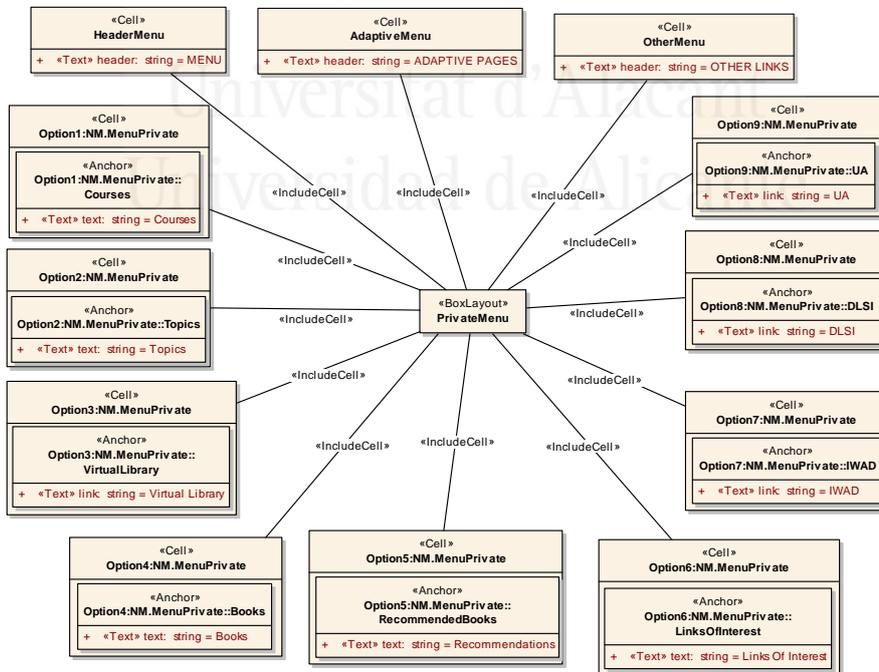


Fig. 6-9. PM for the MenuPrivate page chunk

The final Web page generated on basis of these models is shown in Figure 6-10.



Fig. 6-10 Main Page of the Bibliography System

Note that it is not the objective of the chapter to explain the A-OOH design models in depth or to show the complete set of models. For further information the reader can refer to Chapter 3..

### 6.3.2 Step 2: Adding Personalization using a PRML file

What is left now is defining the Personalization Model in which are specified the adaptive actions to perform over the previously defined set of models. For this purpose we use the PRML language.

Please note that the purpose of this chapter is not to explain the PRML language, for a better understanding of the rules the reader can consult Chapters 4 and 5.

Now we define the PRML configuration file for our running example<sup>5</sup>. For a better comprehension we can divide this file in two sections:

The **acquisition rule section** defines the rules needed to gather the required information to personalize. In our example we have two acquisition rules:

- *The first rule* is triggered by the activation of the link *ViewDetails*. The rule updates the proper instance of the interest class on the author the user consults the details of. This is done using the *SetContent* action.
- *The second rule* is triggered by the start of the session event. The rule stores the device type of the user (*mobile or non mobile device*) in order to personalize.

```

PRMLVersion: Full PersonalizationFrequency: n
# ACQUISITION SECTION
# RULE: "AcquireInterest"
Rule: AcquireInterest Type: Acquisition
When Navigation.BookDetails (NM.Book book) do
Foreach a in (UM.User.userToInterest) do
  If (a.interestToTopic in book.bookToTopic) then
    setContent(a.degree,a.degree+10)
  endIf
endForeach
endWhen

# RULE: "AcquireDeviceContext"
Rule: AcquireDeviceContext Type: Acquisition
When SessionStart do
  setContent
  (UM.User.userToSession.sToMobileDevice.type,
UM.User.userToSession.sToMobileDevice.getDeviceType())
  setContent
  (UM.User.userToSession.sToMobileDevice.name,
UM.User.userToSession.sToMobileDevice.getDeviceName())
endWhen

```

---

<sup>5</sup> Some attributes of the rules have been omitted for simplicity reasons.

The **personalization rule section** contains the personalization rules which describe the effect of personalization in the website. In our example, we have nine personalization rules.

- *The first two rules* are triggered by the *SessionStart* event (i.e. when the user enters the website). They hide the *Recommendations* and *LinksOfInterest* links if there is not enough interest by the user to display them.
- In this website we want to perform the personalization immediately, so to cope with requirements 3<sup>rd</sup> and 4<sup>th</sup> we need two more rules. These rules are triggered by a navigation event to hide or display these links during the navigation of the user (i.e. not only at the start of the session). The third and fourth rules are triggered by the activation of the link *BookDetails*, this means that when the user checks the details of a book these rules will hide or show the corresponding links depending on the user interest on the different topics.
- The *fifth rule* is triggered by the activation of the *Recommendations* link. To fulfil the 1<sup>st</sup> personalization requirement we need to define a *Sort* action. It operates over a set of instances (i.e. the set of books to sort). The syntax of this action is very similar to SQL. This rule properly sorts the book instances by the user interest degree on the different authors returning the fifteen (maximum) with most interest as recommendations to the user.
- The *sixth rule* is triggered by the activation of the *LinksOfInterest* link. To fulfil the 2<sup>nd</sup> requirement this rule selects the proper set of instances to display according with the condition (i.e. interest degree of the user on the different topics greater than a threshold).
- The rest of the personalization rules (7<sup>th</sup>, 8<sup>th</sup>, 9<sup>th</sup>) are triggered by *LoadElement* events. These rules check the device context of the user in order to show or hide the images of the books, book details and recommended books. When the user is browsing the website using a mobile device the books images are not shown for a better readability of the Web pages. This is specified by using the *SelectAttribute* action.

```
# PERSONALIZATION SECTION
# RULE: "HideLink"
Rule: HideInitRecomm Type: Personalization
When SessionStart do
  If ForAll(a in UM.User.userToInterest)
    (a.degree <= 30) then
```

```

        hideLink(NM.RecommendedBooks)
    endIf
endWhen

Rule: HideInitLOI Type: Personalization
When SessionStart do
    If ForAll(a in UM.User.userToInterest)
        (a.degree <= 20) then
            hideLink(NM.LinksOfInterest)
        endIf
    endWhen

Rule: HideRecommendations Type: Personalization
When Navigation.BookDetails (NM.Book book) do
    If ForAll(a in UM.User.userToInterest)
        (a.degree <= 30) then
            hideLink(NM.RecommendedBooks)
        else
            not(hideLink(NM.RecommendedBooks))
        endIf
    endWhen

Rule: HideLinksOfInterest Type: Personalization
When Navigation.BookDetails (NM.Book book) do
    If ForAll(a in UM.User.userToInterest)
        (a.degree <= 20) then
            hideLink(NM.LinksOfInterest)
        else
            not(hideLink(NM.LinksOfInterest))
        endIf
    endWhen

# RULE: "ShowRecommendedBooks"
Rule: RecommendedBooks Type: Personalization
When Navigation.RecommendedBooks(NM.Book* books) do
    sort books orderBy UM.User.userToInterest.degree DES LIMIT 15 Where
    UM.User.userToInterest.degree > 30 and
    UM.User.userToInterest.interestToTopic in books.bookToTopic
endWhen

# RULE: "ShowLinksOfInterest"
Rule: LinksOfInterest Type: Personalization
When Navigation.LinksOfInterest(NM.Topic* topics) do
    Foreach i in (UM.User.userToInterest) do
        If (i.degree > 20) then
            topics.selectInstance(i.interestToTopic)
        endIf
    endForeach
endWhen

# RULE: "HideImageBookDetails"
Rule: HideImageBookDetails Type: Personalization

```

```

When LoadElement.BookDetails(NM.Book book) do
  If (UM.User.userToSession.sToMobileDevice.type = "MobileDevice") then
    not(NM.BookDetails.Attributes.selectAttribute(image))
  endIf
endWhen

# RULE: "HideImageBooks"
Rule: HideImageBooks Type: Personalization
When LoadElement.Books(NM.Book* books) do
  If (UM.User.userToSession.sToMobileDevice.type = "MobileDevice") then
    not(NM.Books.Attributes.selectAttribute(image))
  endIf
endWhen

# RULE: "HideImageRecommend"
Rule: HideImageRecommend Type: Personalization
When LoadElement.Recommendations(NM.Book* books) do
  If (UM.User.userToSession.sToMobileDevice.type = "MobileDevice") then
    not(NM.Recommendations.Attributes.selectAttribute(image))
  endIf
endWhen

```

### 6.3.3 Step 3: Generating the Web Application with AWAC

Once modelled, the Web Application has to be generated using the AWAC tool. The AWAC interface is a Web page in ASP.Net. To generate a Web application using AWAC the following steps are to be taken:

1. Save the UML A-OOH models in XMI format.
2. Create a new project in the AWAC environment and load the XMI files containing the A-OOH models.

This is done in the main view of the AWAC tool, shown in Figure 6-11(a).

The loaded models can be viewed selecting the corresponding option in the *Adaptive OO-H models* section, as it can be seen in Figure 6-11(b).

3. Save the PRML file as a text file and upload the file.

In the menu, the option *PRMLTools* → *Edit Rules* shows a new view of the AWAC tool in which we can load the file containing the PRML rules for our Web application (see Figure 6-12). It is desirable that the extension of the file is “.p” for clarity purposes, but this is not mandatory.

In the same way as in the PRML manager, in the AWAC tool, it is possible to edit the rules loaded and check if they are lexically and syntactically correct (at design time).

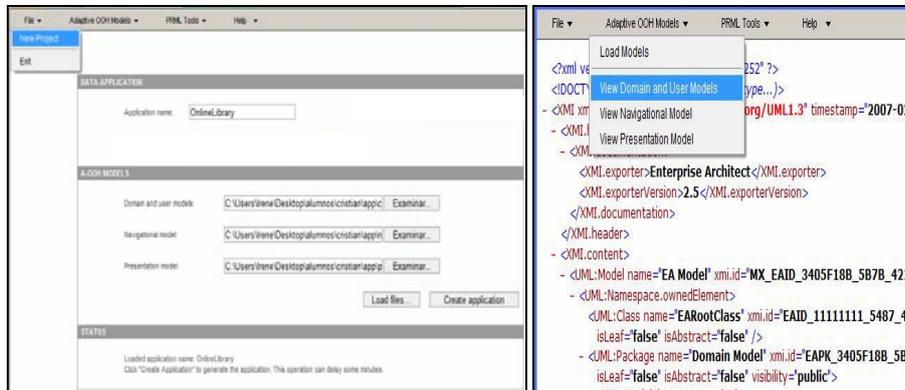


Fig. 6-11. (a) AWAC tool: loading the A-OOH models (b) View of the models in XMI

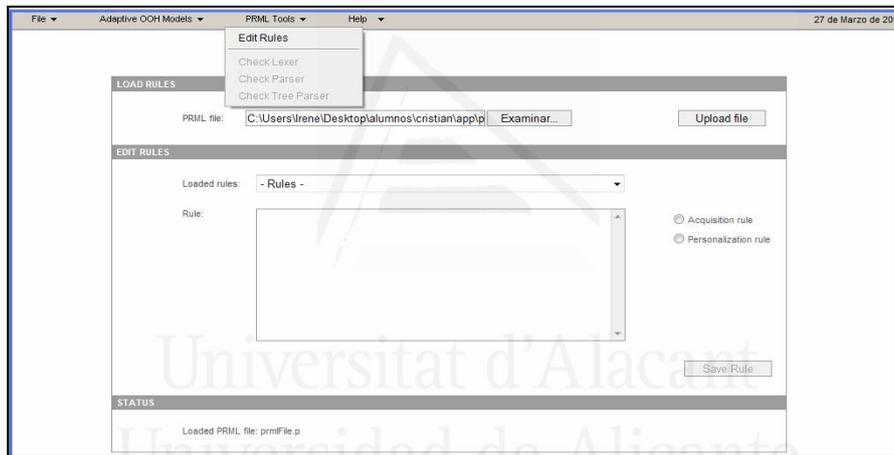


Fig. 6-12. Loading the PRML file

#### 4. Generate and download the Web application

Once the A-OOH models and the PRML file are uploaded, the Web application can be created and downloaded as a compressed rar file (see Figure 6-13).

As explained in Section 6.2, the output of the AWAC tool contains the generated adaptive Website (Web pages in ASP.net), the modules for managing the personalization (the Website Engine, the PRML Evaluator and the PRML Manager) and the application database.



Fig. 6-13. Generate and download the Web application

### 5. *Deploying and running the Website*

Once generated, the adaptive Web application can be run in a Web server. The adaptive Web pages that users have been shown will differ depending on their browsing actions. In Figure 6-14, the *links of interest* page is shown for two different users. Depending on the user interest in the different topics (stored in the UM) the links to recommend vary. To properly recommend links to the user, the AWAC modules generated for managing personalization (explained in Section 6.2) follow the next steps (see Figure 6.2): The *Website Engine* gathers the request of the user for the *links of interest* page and triggers the user browsing event (i.e. click on linksOfInterest) sending it to the *PRML Evaluator* module. This module checks if there is any rule triggered, and finds the "ShowLinksOfInterest" rule which is executed. This rule selects the corresponding links related to the topic instances to be shown from the Domain Model. These recommended may change every time the user clicks on the *LinksOfInterest* link. The option by default is performing the personalization once per session (the recommended link would not change until the next time that the user starts a session. This is the default option and the decision has been taken not to overwhelm the user with constant updates). However, as already explained and for a better and faster testing of the website, in this case study the personalization is performed immediately.

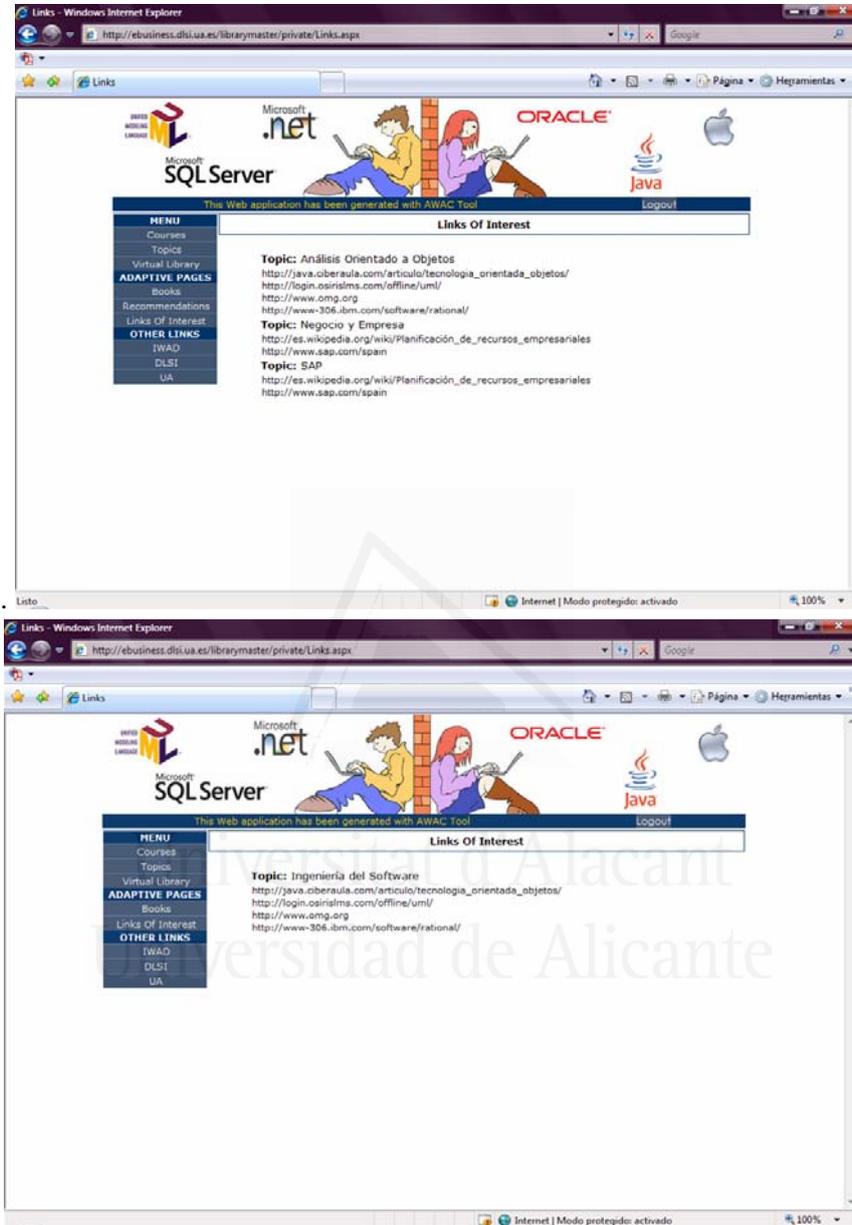


Fig. 6-14. Running the Website: LinksOfInterest page for two different users

## 6.4 Rules Management at Runtime

Once the Web application is generated and deployed with AWAC, let's consider that the designer wants to update a personalization rule at runtime. As described in Section 6.2.2, the PRML Manager allows us to modify the PRML file uploaded for the generated Web application.

To illustrate this fact, let's consider that we want to modify the Rule "HideImageBooks" of our case study. As the reader knows, this rule hides the books images if the user is browsing with a mobile device. It is triggered when the books page is loaded. As explained in Section 6.2.2 we can update the event, condition and action of the rule. We can also modify the frequency of performance of the rules during a session, as explained in Section 6.2.3. Now the value of this attribute is "n". In this case we will not modify it, so the personalization is continuously performed and we can see immediately the result of the adaptation:

```
PRMLversion:Full PersonalizationFrequency: n
```

We will update the condition of the rule.

- Originally, the condition stated that if the user device is mobile the images will be hidden. We will modify this condition now saying that the images will be hidden when the user is browsing the website with a non-mobile device.

We will introduce the modifications into the PRML Manager as shown in Figure 6-15 and if everything is correct the PRML file will be saved.

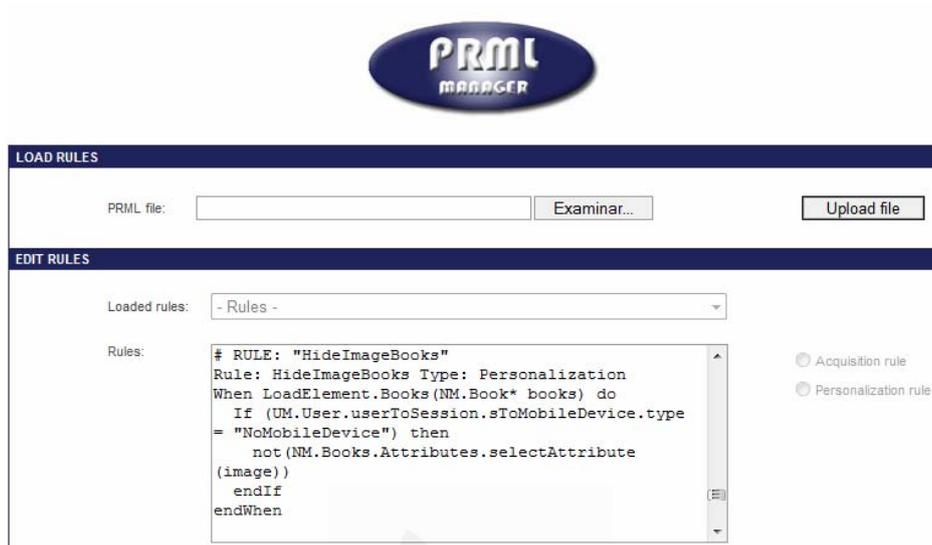


Fig. 6-15. Updating the HideImageBooks rule

Now, the rule should hide the books images when browsing with a PC. So, originally, as we can see in Figure 6-16, the images are not hidden, but when we update the rule, the book images are hidden as can see in Figure 6-17.

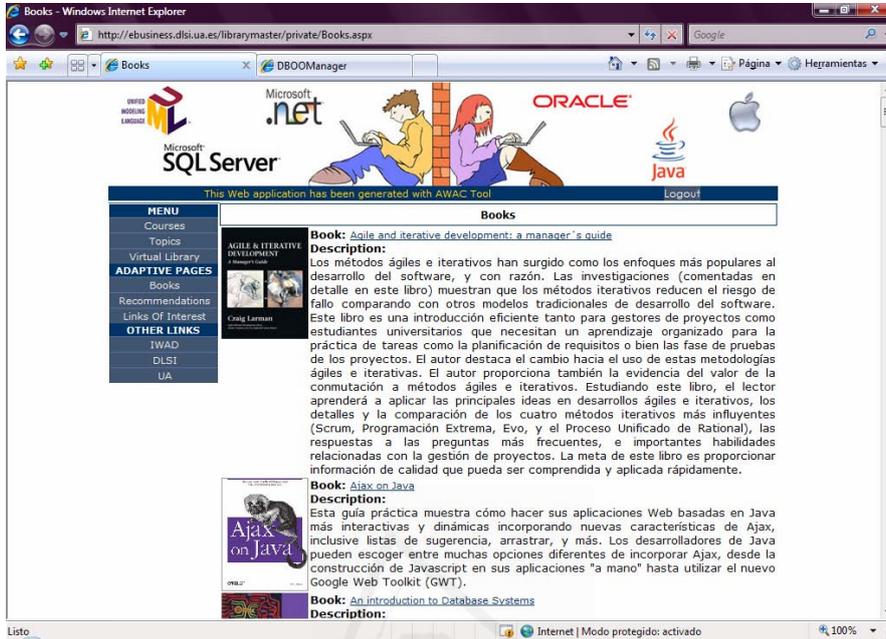


Fig. 6-16. Bibliography System: before updating the rule

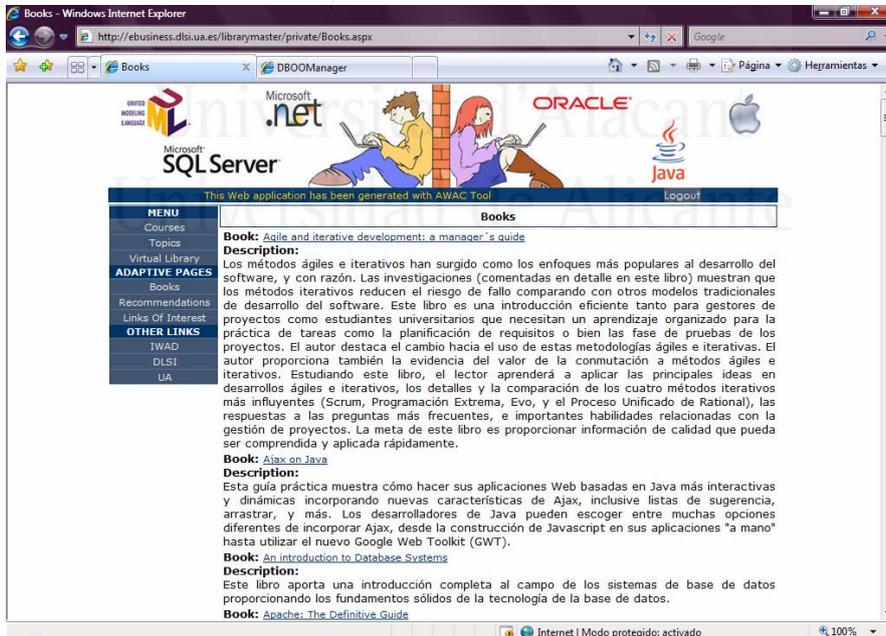


Fig. 6-17. Bibliography System: after the rule update

## 6.5 Results

The main goal of the experiment is to evaluate the satisfaction of the users using the adaptive website. The experimental website ran for two months. The results obtained are described next.

This system is being used in the context of an official master and the students' accesses have been studied, both to the adaptive and non adaptive Web pages. The experiment was first put online *without* personalization support (the rules were all inactive). After 1 month running the number of visits of the different visitors was enough to evaluate the experiment. The next step was putting online the website with the personalization running for another month and comparing the usage of the two websites and the number of accesses to the different parts of the Web applications.

The objective of the experiment is twofold:

- Develop a case study and observe the **suitability** of the chosen implementation in a real world setting.

For this purpose, during several sessions of the master we observed the response rate of the Web pages and the productivity of the students. We can state the response rate is adequate excepting of the accesses to the database, so we should study this issue in depth.

- Study the **browsing behaviour** of the users using the adaptive website

In the next two tables, we can compare the usage of the two websites. Note that the visits refer only to the private part of the websites (i.e. registered users). The first column denotes the different parameters we use to measure the usage of the two websites:

- ***Total visits***

We have measured the total number of visits in the website during one month of usage.

- ***Registered users***

This value represents the number of users which have registered in the website to access the private part of it.

- ***Visits per user***

This parameter represents the number of visits per user. (It is obtained from dividing the total number of visits and the number of registered users)

- ***Pages visited (per user)***

This value contains the average of pages visited per user.

- ***Clicks done***

We also store the total number of clicks done in the website.

**Table 6-2: Non adaptive Website**

Total visits	668
Registered users	114
Visits per user	6
Pages visited (per user)	26
Number of clicks	3020

**Table 6-3: Adaptive Website**

Total visits	1024
Registered users	114
Visits per user	9
Pages visited (per user)	38
Number of clicks	4333

In the tables shown above we can see that the number of visits is more than one third higher in the adaptive website. Due to this, the number of visits per user increases slightly. The number of pages visited in a session is the double in the adaptive website. Looking at these results we could suppose that the users like best the adaptive website. This is probably due to that the number of clicks needed to find the relevant information is reduced. However, to confirm that the increment in the number of clicks was due to the satisfaction of the users, in the adaptive website it has been studied the usage of the different links of the main menu as we can see in the next table.

**Table 6-4: Links usage in the Non-Adaptive and Adaptive Websites**

Links	Usage (%)	
	Non Adaptive	Adaptive
Courses	30,8%	12%
Topics	20%	17%

Virtual Library	16,9%	13,3%
Books	9,2%	10,4%
IWAD	5,4%	3,3%
DLSI	11,5%	5%
UA	6,2%	2,1%
ADAPTIVE PAGES		
Recommendations	-	15,4%
Links of interest	-	21,6%

The first column (*link*) denotes the different links present in main menu of the Web application, the second column represents the media of the percentage of clicks on the different links of the menu of the different users. The links of the menu which have associated a rule that causes personalization are in the section “adaptive pages”. We can conclude the adaptive links situated on the main menu have a high percentage of use in the website relative to the total number of clicks (37%).

To complete this experiment, as short-term work we consider evaluating the satisfaction of the users with questionnaires.

Other experiments are being done with AWAC. One is the generation and running of the (adaptive) Intranet of the university lab of the authors. This Intranet is now online and the users’ accesses are being studied.

## 6.6 Conclusions and Future Work

This chapter presents AWAC, a prototype CAWE tool for the automatic generation of personalized Web applications. This tool implements the A-OOH (Adaptive Object Oriented Hypermedia) methodology, which is an extension of the OO-H method for supporting the modelling of personalized Websites. The input of the AWAC tool is the set of A-OOH design models needed to model the adaptive Website to generate. The output is the generated Website with its database. The output includes a Web engine and a personalization module which allows managing the personalization at runtime of the final Web pages. The personalization rules can be edited in an

independent way of the rest of the application, which improves the personalization maintenance.

We are currently working on an automatic installer. It actually works on Windows XP systems. We are still testing the installer under Windows Vista systems.

As future work, besides implementing all the PRML functionality, we would like to add a graphical user interface in order to define the A-OOH models using AWAC. Now, to define the A-OOH models and generate the XMI files we use the Enterprise Architect Design tool [Enterprise Architect] in which we have defined the UML profiles needed for the modelling of the A-OOH diagrams.



Universitat d'Alacant  
Universidad de Alicante



Universitat d'Alacant  
Universidad de Alicante

## Chapter 7

### PRML: Portability to the UWE approach

In the context of Web modelling methods, personalization strategies are usually tightly intertwined with the rest of the application logic and the underlying technology. This causes problems like the difficulty in maintaining the personalization specifications and the impossibility of reuse of the personalization strategies among different approaches. Moreover, the learning curve of the personalization designer increases when he has to develop adaptive Web applications using different proprietary solutions. As explained in previous chapters (i.e. 4 and 5) PRML solves these problems allowing specifying the personalization as an orthogonal concept. Moreover PRML reduces the learning curve of the personalization designer abstracting him from the programming code, and allowing him to translate a PRML specification to the specifics of a Web methodology.

As explained in Chapter 4, one goal of PRML Lite is that the personalization strategies specified with this language are reusable between different Web modelling approaches. With PRML Full the reusability is limited because of the support of specific and more complex personalization actions (and thus not supported by most of the Web modelling methodologies).

In this dissertation we describe how PRML can be exported to two well-known Web design methods: UWE [Koch, 2001; Koch et al, 2002] and Hera [Houben et al, 2004, Vdovjak et al, 2003]. For this purpose we define transformations from PRML Lite to the specifics of UWE and Hera, which are described along this chapter and Chapter 8 respectively.

The transformations are defined using the Query/View/Transformation (QVT) language [OMG QVT]. This is the standard adopted by OMG to define model transformations. Mappings from PRML specifications to concrete personalization languages were discussed in [Garrigós et al, 2005]. These mappings were defined in an informal way, thus presenting the following

drawbacks: (i) they are difficult to understand, reuse and maintain; and (ii) they have to be manually performed by an expert, which is a tedious and time-consuming task. In order to overcome these problems, we argue for developing these mappings by using QVT. The use of QVT is twofold: on the one hand, mappings are more understandable, formally defined and automatically performed. On the other hand, a repository of mappings can be created, thus making them more maintainable and fully reusable in every project. The aim of this chapter is to formally describe one subset of mappings (transformation rules) of this repository related to the transformation between a PRML personalization rule and a UWE personalization rule [Garrigós et al, 2005; Garrigós et al, 2006].

In the Appendix C the *PRML translator* is described, a prototype tool to automatically implement such transformations from personalization specifications in PRML to the specifics of a personalization specification in a target Web modelling approach. The obtained translation can be used within the CASE tool of the target Web methodology.

The outline of the chapter is as follows: the chapter begins explaining the fundamentals of the QVT language for a better understanding of the transformations. In Section 7.2, the set of QVT relations which define the transformations from PRML to the specifics of the UWE approach are explained. Section 7.3 supplies examples of transformations from PRML to UWE. Finally Section 7.4 presents the conclusions of the chapter.

## 7.1 Query / View / Transformation Language

In this dissertation, to define the needed transformations between PRML and the specifics of the target Web modelling approaches, the QVT language is used. The MOF 2.0 Query/View/Transformation (QVT) language [OMG QVT] is a standard approach for defining formal relations between models. Furthermore QVT is an essential part of the MDA (Model Driven Architecture) standard [OMG, MDA] as a means of defining formal and automatic transformations between models.

QVT consists of two parts: declarative and imperative. The declarative part provides mechanisms to define relations that must hold between the model elements of a set of candidate models (source and target models). This declarative part can be split into two layers according to the level of abstraction: the relational layer that provides graphical and textual notation for a declarative specification of relations, and the core layer that provides a simpler, but verbose, way of defining relations. The imperative part defines operational mappings that extend the declarative part with imperative implementations when it is difficult to provide a purely declarative specification of a relation.

In this work, we focus on the relational layer of QVT. This layer supports the specification of relationships that must hold between models by means of a relations language. A relation is defined by the following elements:

- **Two or more domains:** each domain is a set of elements related to a source or a target model. The kind of relation between domains must be specified: checkonly (C), i.e., it is only checked if the relation holds or not; and enforced (E), i.e., the target model can be modified to satisfy the relation.
- **When clause:** it specifies the conditions under which the relation needs to hold (i.e. precondition).
- **Where clause:** it specifies the condition that must be satisfied by all model elements participating in the relation (i.e. postcondition).

Defining relations by using the QVT language has the following advantages: it is a standard language, (ii) relations are formally established, easily understandable, and automatically performed, and (iii) relations can be easily integrated in an MDA approach.

## 7.2 Transformations from PRML to the specifics of UWE

In this section the transformations from PRML to the specifics of the UWE approach are defined. Following the QVT standard, we have developed a set

of relations to transform a PRML rule into an OCL expression that represents the same rule according to the UWE approach. This set of relations is explained along the next sections. To properly define these transformations we need to know both metamodels (PRML Lite and UWE). The PRML Lite metamodel was explained in Chapter 4. Next we give an overview of the UWE adaptation metamodel.

### 7.2.1 The UWE Adaptation metamodel

As explained in Chapter 2 (see *Section 2.3.4* for details), UWE provides a reference model for adaptive hypermedia systems (Munich reference model), based on the Dexter reference model [Van Ossenbruggen and Eliëns, 1995]. The purpose of this model is to identify the main features of adaptive hypermedia and personalized Web applications, as a prior step to the definition of appropriate modelling techniques.

The Adaptation Model consists of containers for user behaviour, a set of rules and a set of functions to perform the adaptation functionality. A rule is modelled as a class Rule that consists of one Condition one Action and attributes. Rules are expressed as OCL expressions in the context of the `executor()` method of the rule. In Figure 7-1 the UWE adaptation metamodel is shown. Figure 7-2 shows the OCL metamodel for expressions. For the complete OCL metamodel see [UML 2.0 OCL specification].

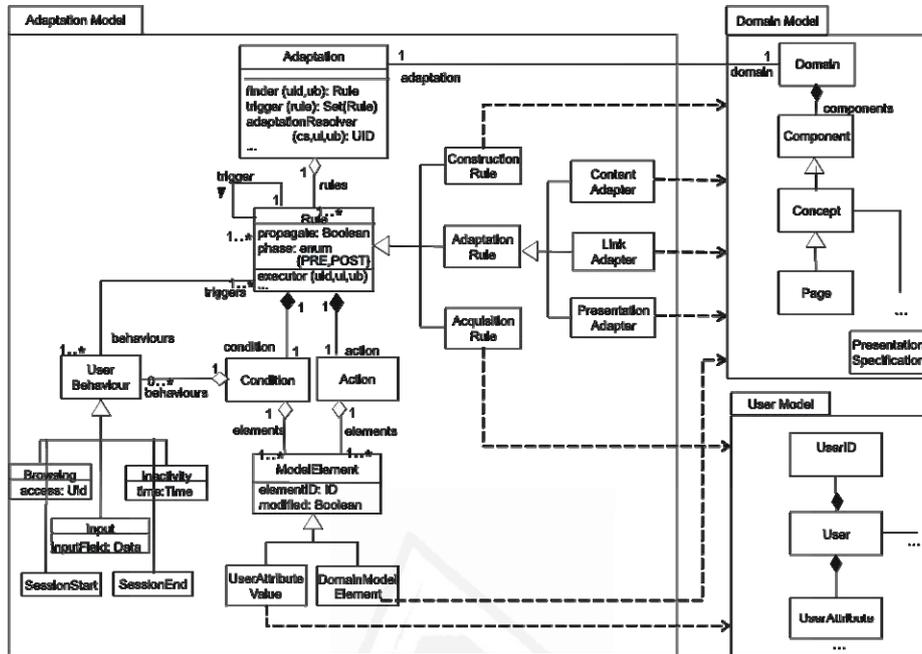


Figure 7-1 UWE Adaptation Metamodel

### 7.2.2 Sequence of QVT Relations

In this section we present the QVT relations to transform a PRML rule into an OCL expression that represents the same personalization rule for the UWE approach. In order to apply the QVT relations, a model of each PRML rule must be defined according to the PRML Lite metamodel (presented in Chapter 4, Figure 4-3). The output of a QVT transformation is an OCL expression based on the abstract syntax defined in the specification [UML 2.0 OCL specification]. Furthermore, in this specification it is also defined a mapping between this abstract syntax and the concrete syntax used for defined OCL expressions in textual notation. Therefore it is quite straightforward to obtain a model from PRML and OCL expressions useful for applying the QVT transformations, and vice versa, therefore, in this dissertation we focus on presenting both PRML and UWE rules in textual notation, although the QVT relations have been defined over their corresponding metamodels (PRML and OCL).

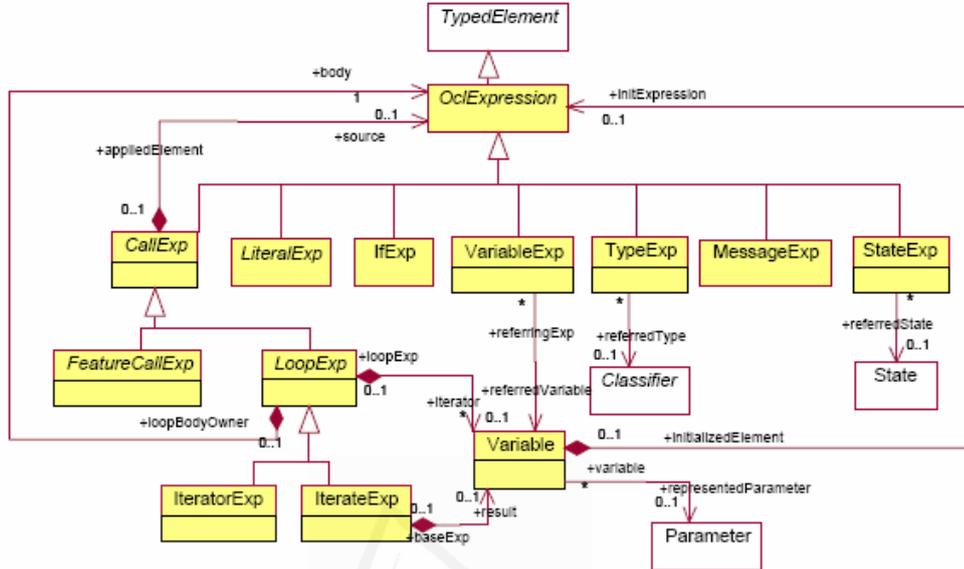


Figure 7-2: OCL Metamodel for expressions

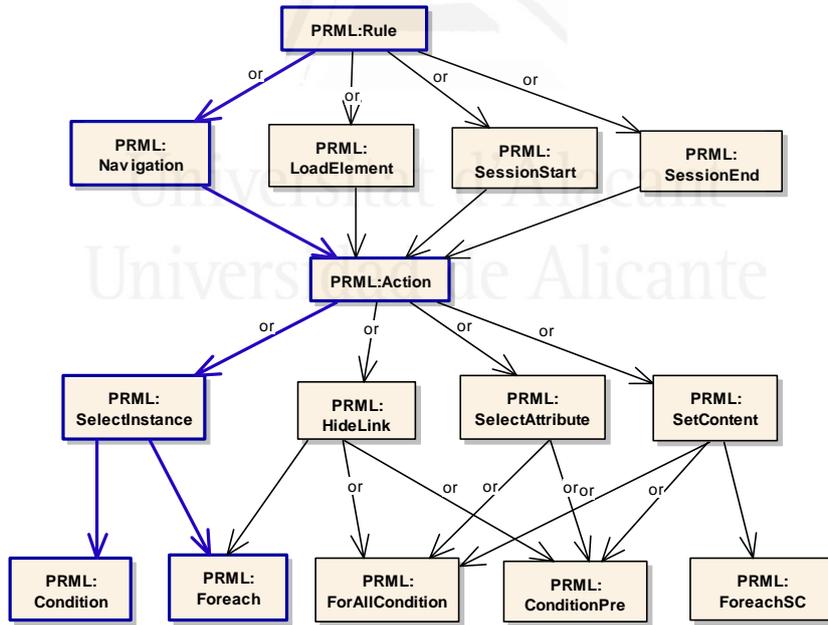


Figure 7-3 Sequence of Transformations

In Figure 7-3 we can see the sequence of the transformations that needs to be performed for the different types of PRML rules. This set of mappings is applied according to the structure of a PRML rule (see section 4.2 of Chapter 4): first the *PRMLRule* relation is applied to drive the whole transformation. In this way, this rule decides about which is the following mapping to be applied (according to the expression defined in the “where” clause). If the event is *Navigation*, then the *PRMLNavigation* relation is applied. The *PRMLNavigation* relation has to call to the *PRMLAction* relation, since the main part of a PRML rule is the Action to be performed. The *PRMLAction* relation deals with executing the right relation according to the kind of Action. Then, if the Action is for example *SelectInstance*, the *PRMLSelectInstance* relation is applied. This relation together with *PRMLCondition* and *PRMLForEach* relations are used for creating the corresponding OCL model from the PRML model.

This set of QVT relations defined for transforming PRML to the specifics of the UWE approach are explained in detail along the next sections.

### 7.2.2.1 PRMLRule Relation

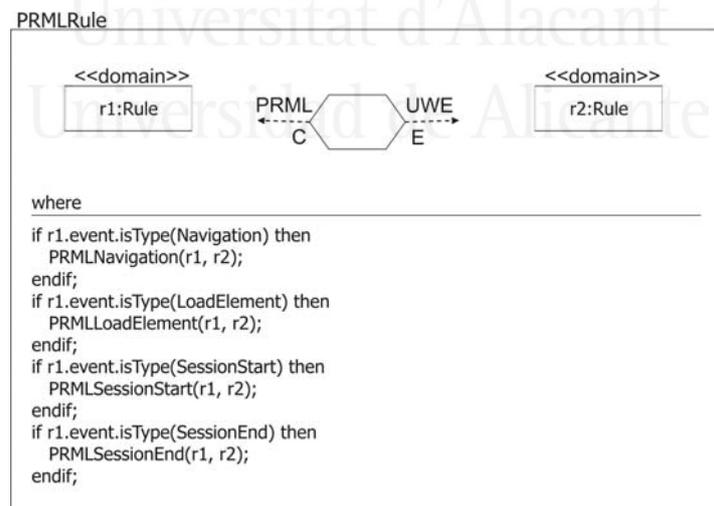


Figure 7-4 PRMLRule relation

A PRML rule is transformed into an UWE rule. The “where” clause checks the event type of the PRML rule calling the appropriate QVT relation.

### 7.2.2.2 Events Relations

There are four QVT relations defined for specifying the transformations from the different types of events in PRML Lite: *PRMLNavigation*, *PRMLLoadElement*, *PRMLSessionStart* and *PRMLSessionEnd*. The transformations from PRML events to UWE events are straightforward, as UWE has equivalent events defined in the adaptation metamodel (see figure 7-1).

#### 7.2.2.2.1 PRMLNavigation Relation

A PRML *navigation* event is transformed into a UWE *user browsing* event. The “where” clause contains a relation (*PRMLAction*) that extends the previous one. This relation is described in Section 7.2.5.

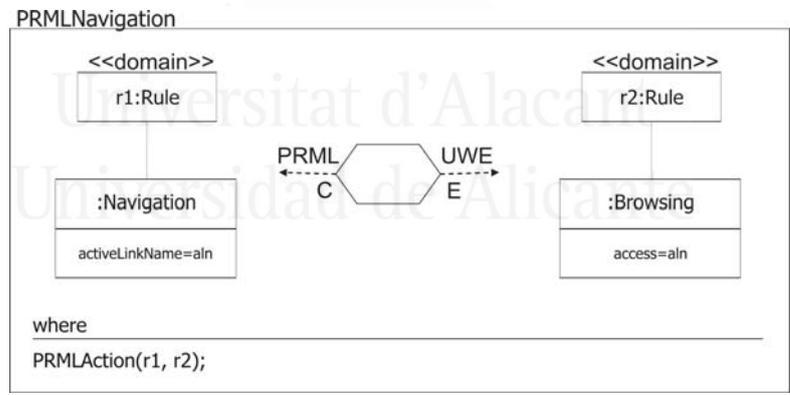


Figure 7-5 PRMLNavigation relation

#### 7.2.2.2.2 PRMLLoadElement Relation

If the PRML rule is not triggered by a *navigation* event but by a *loadElement* event this relation is applied. A PRML *loadElement* event is transformed into a UWE *user browsing* event. The difference with the previous relation is that

in this case the event is caused when a node is loaded (independently of the link which loaded it). This relation is extended by the *PRMLAction* relation as we can see in the “where” clause.

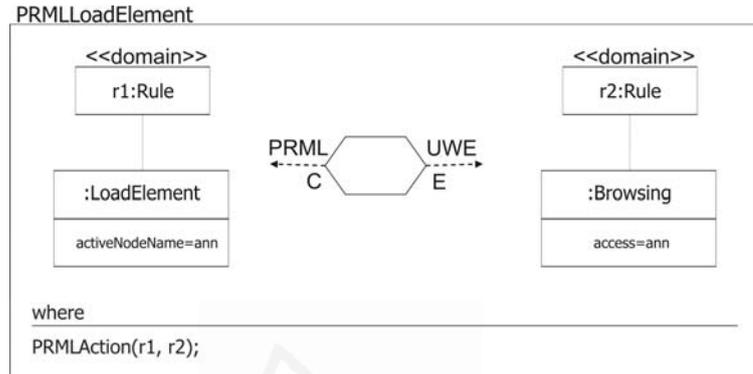
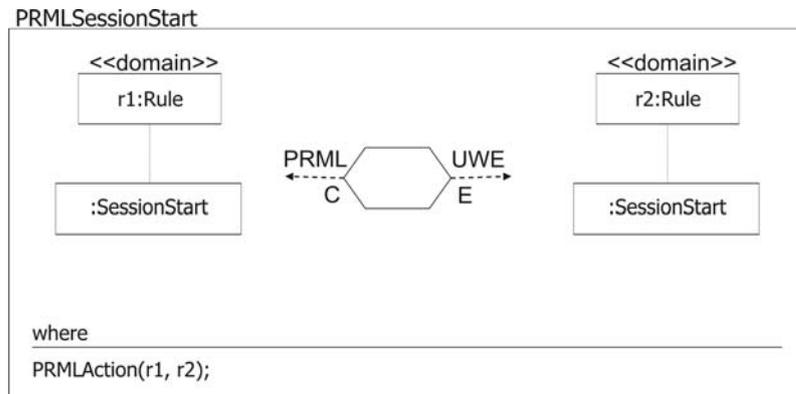


Figure 7-6 PRMLLoadElement relation

### 7.2.2.2.3 PRMLSessionStart Relation

If the PRML rule is triggered by a *SessionStart* event the *PRMLSessionStart* relation is applied. A PRML *SessionStart* event is transformed into a UWE *SessionStart* event. This transformation is straightforward. The same as the other event relations, this relation is extended by the *PRMLAction* relation in the “where” clause.





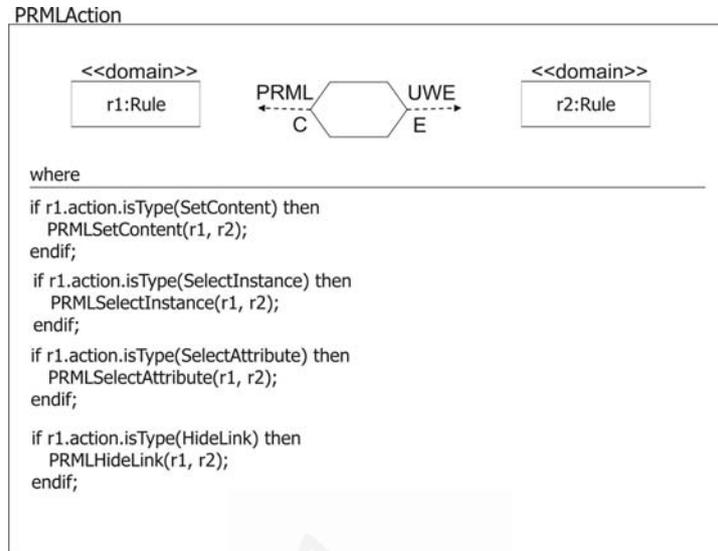


Figure 7-7 PRMLAction relation

After applying the *PRMLRule* relation and one of the events relations defined (depending on the event that triggers the PRML rule to be transformed), the *PRMLAction* relation checks the kind of action in the PRML rule in order to call the adequate relation in the “where” clause.

#### 7.2.2.4 Actions Relations

A PRML action is transformed into an OCL expression that represents the UWE rule. Depending on the type of action this OCL expression is built differently. There are four QVT relations defined for specifying the transformations from the different types of actions in PRML Lite: *PRMLSelectInstance*, *PRMLSelectAttribute*, *PRMLHideLink* and *PRMLSetcontent*.

##### 7.2.2.4.1 *PRMLSelectInstance* Relation

This relation checks that there is a set of elements in the PRML rule that represents a *SelectInstance* action according to the PRML Lite metamodel

(see Fig. 4-3). These elements are: a *SelectInstance* class together with the corresponding Variable (that represents a collection) and the Type of this Variable.

This relation enforces that the corresponding OCL expression has the following elements: an *IteratorExp* class (whose type is “Select”), a collection of elements (that represents the source of the IteratorExp) of the same type of the Variable of the PRML *SelectInstance*.

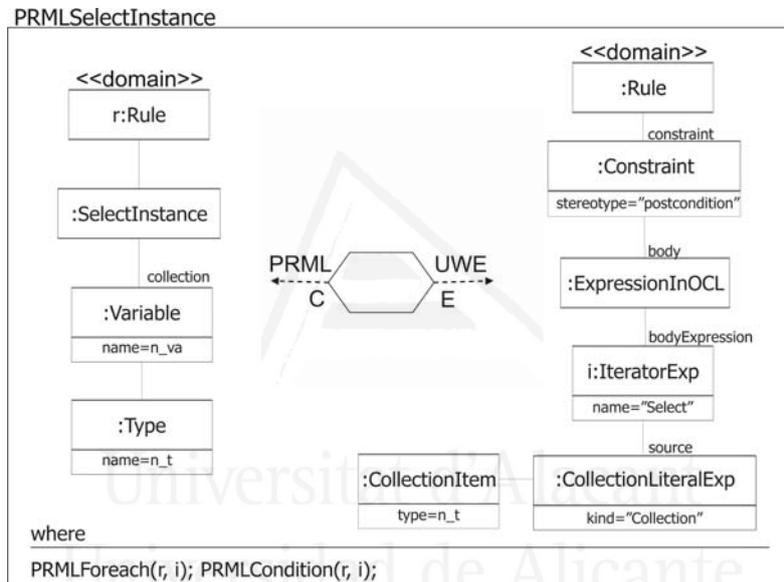


Figure 7-8 PRMLSelectInstance relation

Once this relation holds, the *PRMLForEach* and the *PRMLCondition* relations must be performed, according to the “where” clause. In this way, every part of a PRML rule will be checked with its corresponding QVT rule, and the corresponding elements in the OCL expression will be created. These relations are explained in the Section 7.2.7.1 and 7.2.7.3 respectively.

#### 7.2.2.4.2 PRMLSelectAttribute Relation

This relation checks that there exists a set of elements in the PRML rule that represents a *SelectAttribute* action according to the PRML Lite metamodel

(see Fig. 4-3). These elements are: a *SelectAttribute* class together with the corresponding *Variable* (that represents a collection) and the *Type* of this *Variable*, and a *ModelElement* which represents the attribute to be selected.

This relation enforces that the corresponding OCL expression has the following elements: an *IteratorExp* class (whose type is “Select”), a collection of elements (that represents the source of the *IteratorExp*) of the type “Attributes” representing the collection of attributes of the *Variable* of the PRML *SelectAttribute* element. The body of the *IteratorExp* is an *OCLExpression* which checks the name of the attribute coincides with the name of the PRML *ModelElement* representing the attribute to be selected. Finally, the *iterator* of the *IteratorExp* contains the definition of a variable of the type “Attribute”.

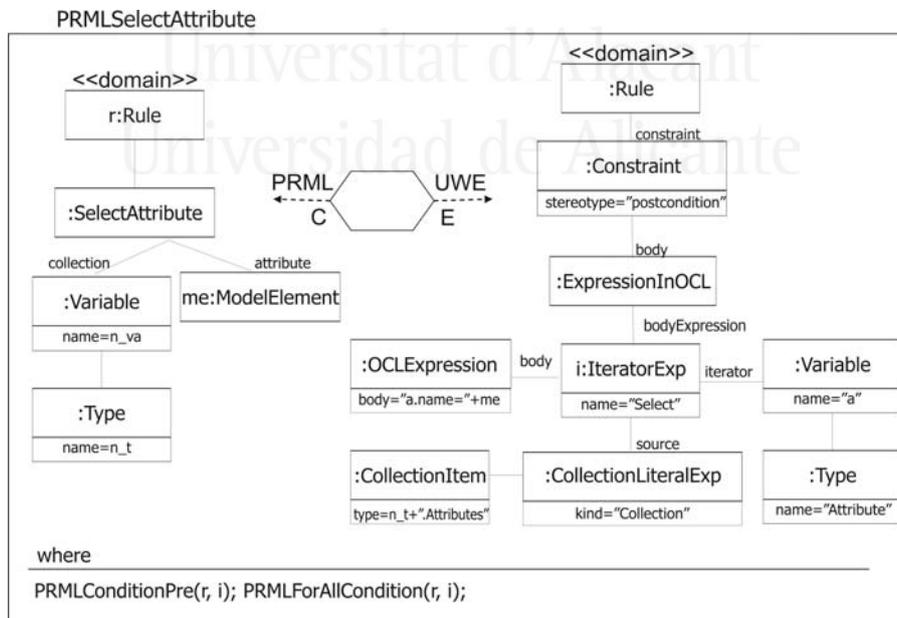


Figure 7-9 PRMLSelectAttribute relation

Once this relation holds, the *PRMLConditionPre* and the *PRMLForAllCondition* relations must be performed, according to the “where” clause. In this way, every part of a PRML rule will be checked with its corresponding QVT rule, and the corresponding elements in the OCL expression will be created. These relations are explained in the Section 7.2.7.4 and 7.2.7.5 respectively.

#### 7.2.2.4.3 *PRMLHideLink Relation*

This relation checks that there is a set of elements in the PRML rule that represents a *HideLink* action according to the PRML Lite metamodel (see Fig. 4-3). These elements are: a *HideLink* class together with a *ModelElement* which represents the link to be hidden.

This relation enforces that the corresponding OCL expression has the following elements: an *IteratorExp* class (whose type is “Reject”), a *StringLiteralExpression* (that represents the source of the *IteratorExp*) representing all the links instances of the model. The body of the *IteratorExp* is an *OCLExpression* which checks that the UID of the link coincides with the PRML *ModelElement* name. Finally, the *iterator* of the *IteratorExp* contains the definition of a variable of the type “Link”.

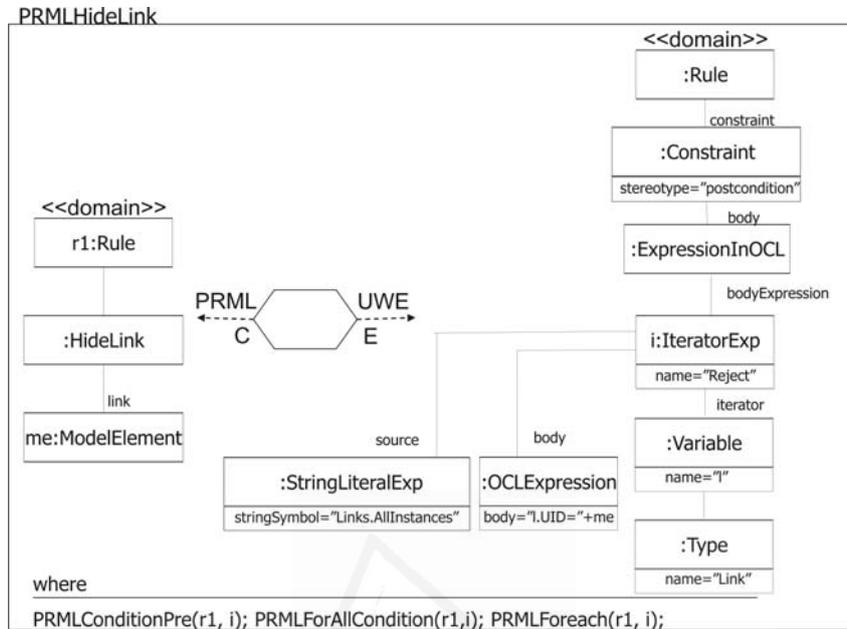


Figure 7-10 PRMLHideLink relation

Once this relation holds, the *PRMLConditionPre*, the *PRMLForAllcondition* and the *PRMLForEach* relations must be performed, according to the “where” clause. In this way, every part of a PRML rule will be checked with its corresponding QVT rule, and the corresponding elements in the OCL expression will be created. These relations are explained in the Section 7.2.7.4, 7.2.7.5 and 7.2.7.1 respectively.

#### 7.2.2.4.4 PRMLSetContent Relation

This relation checks that there is a set of elements in the PRML rule that represents a *SetContent* action according to the PRML Lite metamodel (see Fig. 4-3). These elements are: a *SetContent* class together with a *ModelElement* which represents the attribute to be updated.

This relation enforces that the corresponding OCL expression has the following elements: an *ExpressionInOCL* class which body is the expression or value given in the PRML *SetContent* element. The value of the expression

depends on if the attribute to be updated is involved in the OCL expression because in OCL in those cases the *@pre* statement is used.

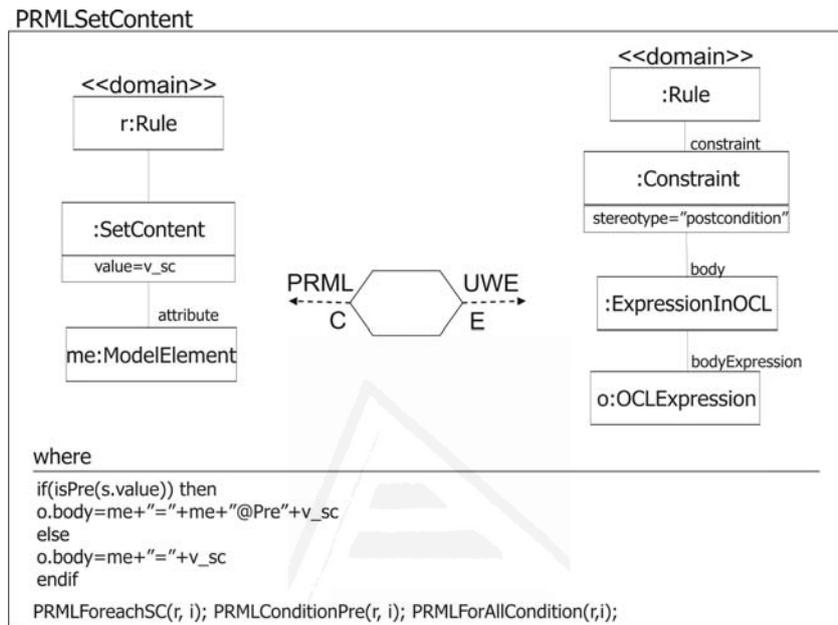


Figure 7-11 PRMLSetContent relation

Once this relation holds, the *PRMLForEachSC*, the *PRMLConditionPre* and the *PRMLForAllCondition* relations must be performed, according to the “where” clause. In this way, every part of a PRML rule will be checked with its corresponding QVT rule, and the corresponding elements in the OCL expression will be created. These relations are explained in the Sections 7.2.7.2, 7.2.7.4 and 7.2.7.5 respectively.

### 7.2.2.5 Foreach and Condition Relations

There are different relations for mapping the PRML foreach and condition expressions. This is because depending on the type of action performed it is mapped differently.

7.2.2.5.1 PRMLForEach Relation

This relation checks that there is the following set of elements in the PRML rule: a *ForEach* element, with a *Variable* which has a *Type* and a *PathExpression*. This relation enforces that the OCL expression contains an *IteratorExpression* which has a *Variable* (with its *Type*). This *Variable* is the iterator of the *IteratorExp* element.

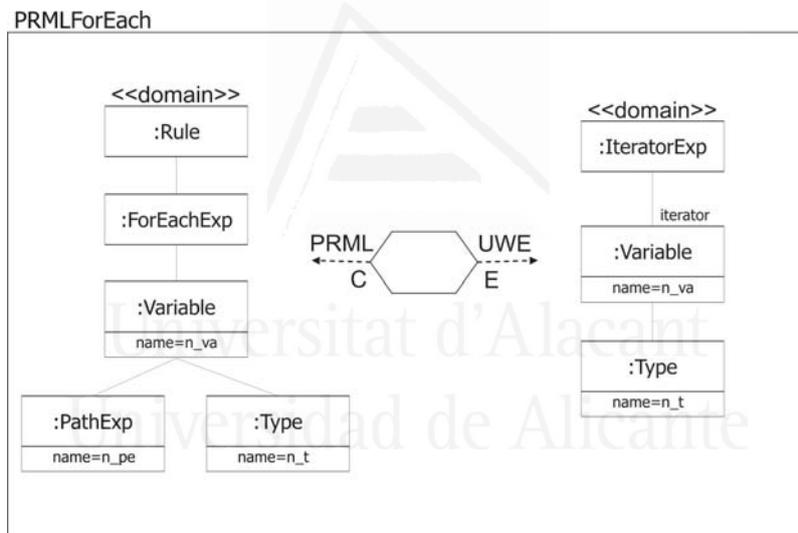


Figure 7-12 PRMLForEach relation

7.2.2.5.2 PRMLForeachSC Relation

The *ForEach* expression is mapped differently when the action of the rule is *SetContent*. This relation checks that there is the following set of elements in the PRML rule: a *ForEach* element, and an attribute (*ModelElement*). This

relation enforces that the OCL expression contains a Constraint that is a *precondition* with an *IteratorExpression* of the type *Exists*, which has as the source a collection of the same type as the PRML attribute.

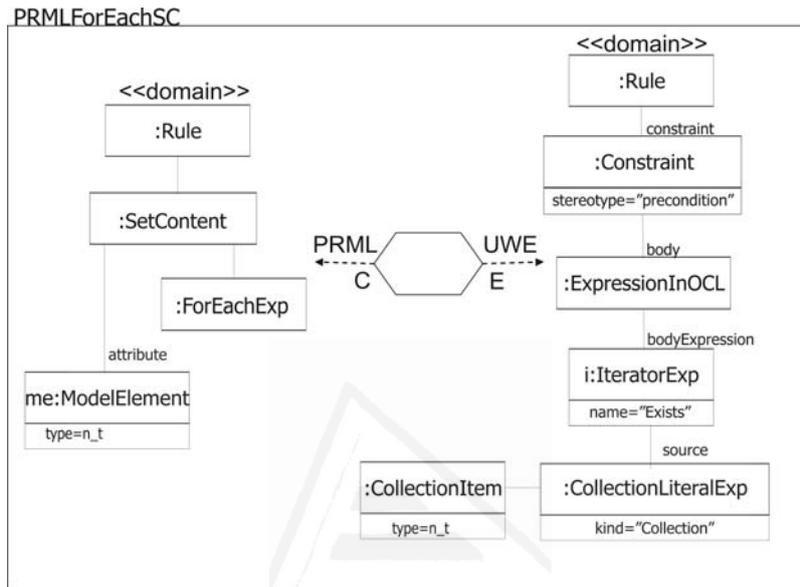


Figure 7-12 PRMLForEachSC relation

### 7.2.2.5.3 PRMLCondition Relation

This relation checks the *Condition* class together with its boolean expression (*BoolExp*) within the PRML rule. These elements enforce the body of the *IteratorExp*, as a OCL expression that corresponds with the boolean expression of the *Condition* element in the PRML rule.

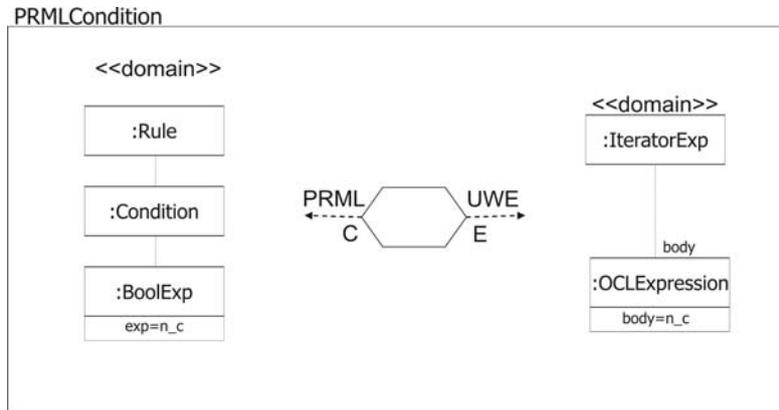


Figure 7-13 PRMLCondition relation

#### 7.2.2.5.4 PRMLConditionPre Relation

This relation checks the *Condition* class together with its boolean expression (*BoolExp*) within the PRML rule. The *when* clause checks that the forall expression should be null for this relation. These elements enforce a *precondition* with an OCL expression that corresponds with the boolean expression of the *Condition* element in the PRML rule.

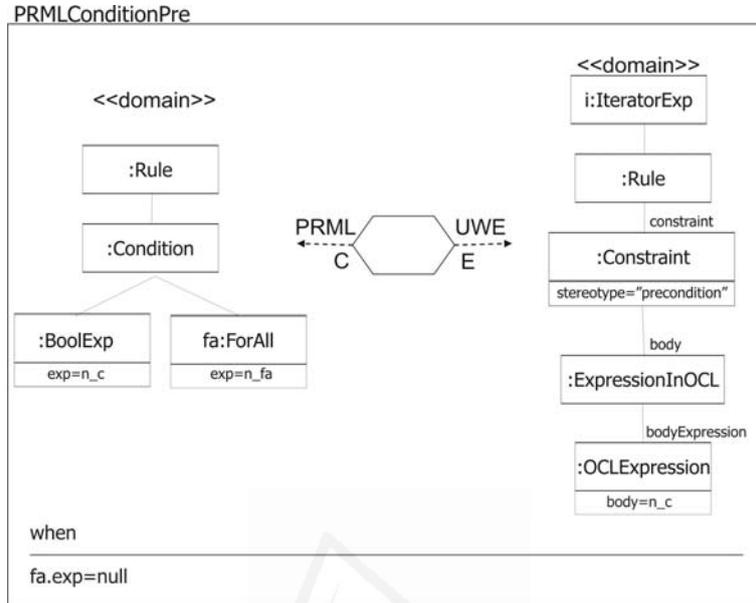


Figure 7-14 PRMLConditionPre relation

7.2.2.5.5 PRMLForAllCondition Relation

This relation checks the *Condition* class together with its boolean expression (*BoolExp*) within the PRML rule. The *when* clause checks that a *forall* expression should exist. These elements enforce a *precondition* with an OCL *forall* Iterator expression. The body of the Iterator expression corresponds with the boolean expression of the Condition element in the PRML rule.

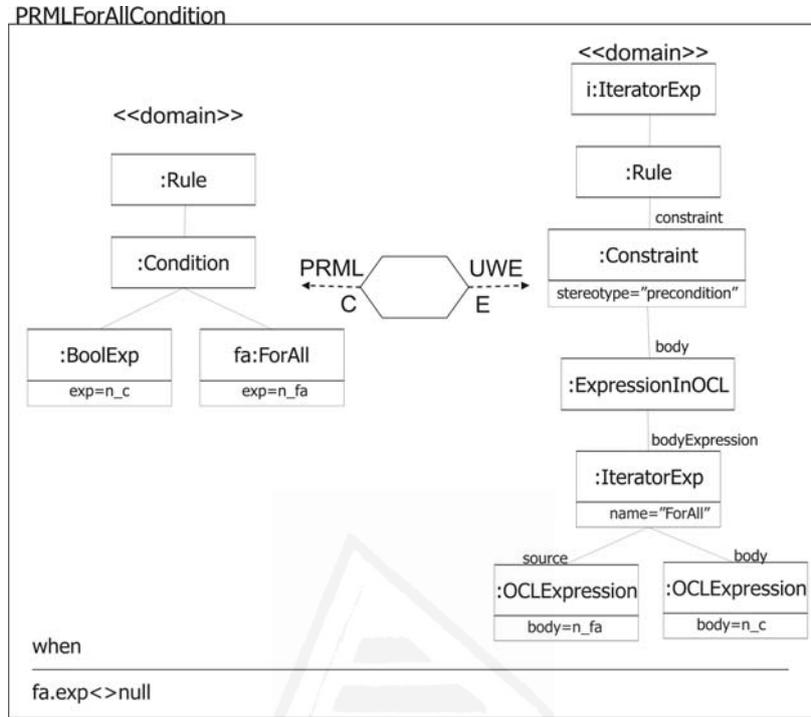


Figure 7-15 PRMLForAllCondition relation

### 7.2.3 Process Transformation Example

As an example, let's see how is performed the transformation needed for the following PRML rule. This rule was defined in section 4.3.3 of Chapter 4 to fulfil the 2nd personalization requirement described in the running example (i.e. offer recommendations based on the user interest on authors). This rule selects the instances (to be displayed) of the book concept which are written by an author in which the user has interest degree greater than 100.

```

When Navigation.Recommendations(NM.Book* books) do
  Foreach b,a in (books,UM.User.usertoInterest.interestToAuthor)
  do
    If (UM.User.userToInterest.degree > 100 and
    a.ID =b.booksToAuthor.ID) then
      books->selectInstance(b)
    endIf
  endForeach

```

```
endWhen
```

In UWE the corresponding rule is next:

```
Context: Rule::Executor( uid:ComponentUID, ui:UserID, ub:UserBehaviour )
Post: result=Collect(NM.Book)->Select(b:NM.Book,
a:UM.User.userToInterest.interestToAuthor | b.bookToAuthor.ID=a.ID
and UM.User.userToInterest.degree>100)
```

In the postcondition of the rule, a Select OCL expression selects the instances of the book in which the user has an interest degree greater than 100.

In this rule we can see how the QVT relations described in this section are applied:

- **PRMLRule:** A PRML rule is transformed into an UWE rule.
- **PRMLNavigation:** A PRML navigation event is transformed into a user browsing event. This transformation cannot be seen in the textual representation because it directly affects the UWE adaptation metamodel.
- **PRMLAction:** A PRML action is transformed into an OCL expression that represents the UWE rule.
- **PRML SelectInstance:** Once checked that the elements representing the *SelectInstance* action exist in the PRML rule the relation enforces that the corresponding OCL expression has the following elements: an *IteratorExp* class (whose type is “Select”), a collection of elements (that represents the source of the *IteratorExp*) of the same type of the Variable of the PRML *SelectInstance*.

```
Post: result=Collection(NM.Restaurant)->Select()
```

- **PRMLForEach:** The *ForEach* element (with a Variable which has a Type and a *PathExp*) of the PRML rule is transformed to an *IteratorExp* which has a Variable (with its Type). This Variable is the iterator of the *IteratorExp*

```
Post: result=Collection(NM.Restaurant)->Select(b:NM.Restaurant,
i:UM.User.uToBookings.bToBookingList | )
```

- PRMLCondition:** The body of the IteratorExp is defined as an OCL expression that correspond with the boolean expression of the Condition element in the PRML rule.

```
Post: result=Collection(NM.Restaurant)->Select(b:NM.Restaurant,
i:UM.User.uToBookings.bToBookingList | b >= 5 and a.category =
b.bLToRestaurant.category)
```

Next, examples of transformations using the different types of actions are shown.

### 7.3 PRML To UWE: Transformation Examples

The examples shown in this section refer to the different actions of PRML, which are based in the running example described in Chapter 4 (i.e. an online library). The corresponding UWE models (i.e. domain and user model and navigation model) are shown next. Note that is not the purpose to explain in detail these models but just help in the understanding of the examples.

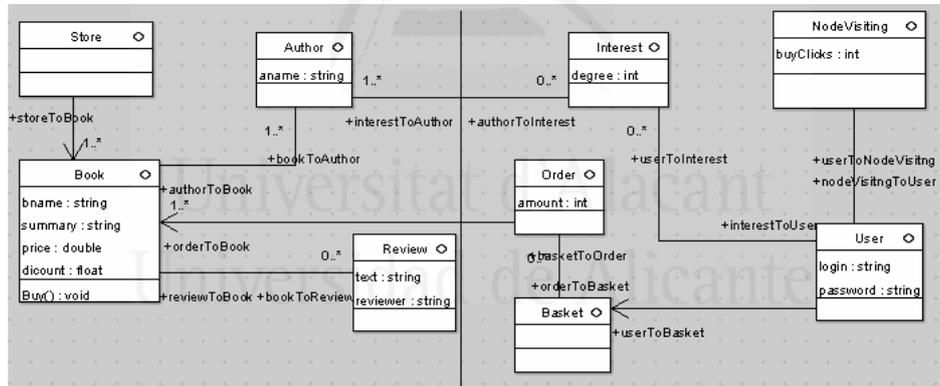


Figure 7-16 DM-UM for UWE

The DM and UM are represented by a class diagram. Figure 7.16 shows the NM for the UWE approach for the case study of the dissertation presented in Chapter 4. For more details on the UWE models NM consult [Koch, 2001].

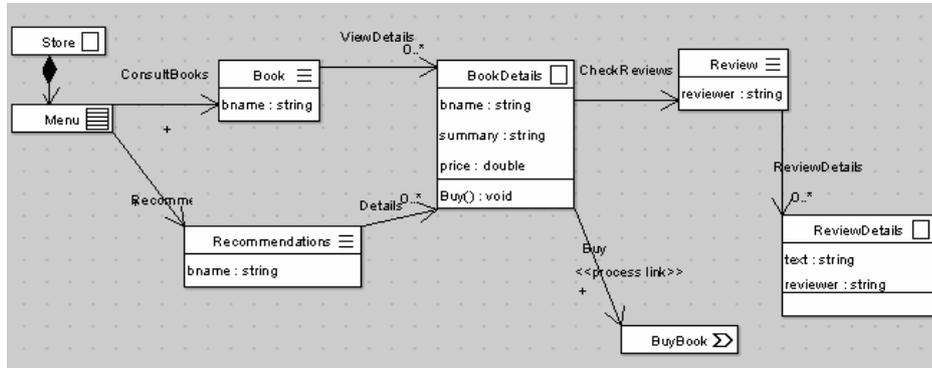


Figure 7-17 UWE Navigation Diagram

To fulfill the personalization requirement described in previous section (users will see recommendations of books based on their interests in authors) we need to store the interest of the user in the User Model as described before. We consider that the user is interested in an author when he consults the details of a book of this author. The PRML rule updating the interest of the user in the User Model is explained in Section 4.3.1, its counterpart in UWE is next:

- **SetContent Example**

<b>PRML</b>	<pre> <b>When</b> Navigation.ViewDetails (NM.Book b) <b>do</b>   <b>Foreach</b> i <b>in</b> (UM.User.userToInterest) <b>do</b>     <b>If</b> (b.bookToAuthor.ID = i.interestToAuthor.ID) <b>then</b>       <b>setContent</b>(i.degree, i.degree+10)     <b>endIf</b>   <b>endForeach</b> <b>endWhen</b> </pre>
<b>UWE</b>	<pre> <b>Context:</b> Rule::Executor(uid:ComponentUID, ui:UserID, ub:UserBehaviour) <b>Pre:</b> (UM.User.userToInterest) -&gt;<b>exists</b> (i:Interest  NM.Book.bToAuthor.ID=i.iToAuthor.ID) <b>Post:</b> <b>result</b>= i.degree = i.degree@Pre+10 </pre>

- The precondition of the rule corresponds with the *foreach* and *condition* statements. Here is checked if there exists one instance of the interest class

(related with the author class) that coincides with the book the user is consulting.

- In the postcondition of the rule, the interest degree is updated.
- The *navigation* event is mapped to a UWE browsing event but this is not textually represented in the OCL expression.

- **SelectAttribute Example**

To illustrate the SelectAttribute action of PRML Lite, we consider the 1<sup>st</sup>: personalization requirement defined in the case study: When the user clicks on the *ViewDetails* link to consult a specific book, if the number of books bought is greater than 10, we show an attribute “discount” to the user (this rule was explained in the Section 4.3.2 of Chapter 4).

<b>PRML</b>	<pre> When Navigation.ViewDetails(NM.BookDetails book) do   If UM.User.uToNodeVisiting.buyClicks &gt;10 then     book.Attributes.selectAttribute(discount)   endIf endWhen         </pre>
<b>UWE</b>	<pre> Context: Rule::Executor(uid:ComponentUID,ui:UserID,ub:UserBehaviour) Pre: UM.User.uToNodeVisiting.buyClicks&gt;10 Post: result=book.Attributes-&gt;Select(a:Book.attribute   a.name="discount")         </pre>

- In the precondition of the rule, the condition over the number of clicks is described.
- In the postcondition of the rule, a Select OCL expression selects the attribute “discount” from the collection of attributes of the book concept.
- The *navigation* event is mapped to a UWE browsing event but this is not textually represented in the OCL expression.

- **HideLink Example**

To illustrate the HideLink action of PRML Lite, we consider the 3rd requirement of the case study of this dissertation: If the user interest degree on the different authors is not initialized (this means the user has not still consulted the details of any book, maybe is the first time the user goes into the website) the *recommendations* link is not shown. Note that there is only one *Recommendations* link in the website. The PRML rule was explained in the Section 4.3.4 of the Chapter 4. The UWE rule is next:

<b>PRML</b>	<pre> When SessionStart do   If ForAll(UM.User.usertoInterest)     (UM.User.usertoInterest.degree=null or     UM.User.usertoInterest.degree&lt;100) then     hideLink(NM.Recommendations)   endif endWhen </pre>
<b>UWE</b>	<pre> Context: Rule::Executor(uid:ComponentUID,ui:UserID,ub:UserBehaviour) Pre: (UM.User.usertoInterest)- &gt;forAll(UM.User.usertoInterest.degree=null or UM.User.usertoInterest.degree&lt;100) Post: result=Links.AllInstances-&gt;Reject (l:link   l.UID="Recommendations") </pre>

- In the precondition of the rule, a condition over all the instances of the interest concept must be satisfied: the interest degree should not be initialized.
- In the postcondition of the rule, a Reject OCL expression selects the link “Recommendations” from the collection of links, to be hidden.
- The *SessionStart* event is mapped to a UWE *sessionStart* event but this is not textually represented in the OCL expression.

## 7.4 Conclusions

In this dissertation we argue that PRML Lite can be exported to the specifics of different Web methodologies. This implies that the PRML language is reusable as well as the personalization strategies defined with it.

In this chapter it has been explained how PRML Lite can be exported to the specifics of the UWE approach defining a set of transformations to conceptually show it. These transformations are defined using the QVT language. This chapter shows the set of relations needed for defining such transformations supplying examples to better illustrate them.

The Appendix C of this dissertation describes the *PRML translator*, a prototype tool that automatically implements the defined transformations. Next chapter presents the set of transformations for another well-known Web methodology: Hera.



Universitat d'Alacant  
Universidad de Alicante



Universitat d'Alacant  
Universidad de Alicante

## Chapter 8

### PRML: Portability to the Hera approach

In this chapter the transformations from PRML to the Hera approach are described. The same as in the previous chapter, the transformations are defined using the Query/View/Transformation (QVT) language [OMG QVT] which is the standard adopted by OMG to define model transformations. The aim of this chapter is to formally describe one subset of mappings (transformation rules) related to the transformation between a PRML personalization rule and a Hera personalization rule [Garrigós et al, 2005; Garrigós et al, 2006].

The outline of the chapter is as follows: Section 8.1 describes the set of QVT relations which define the transformations from PRML to the specifics of the Hera approach. Section 8.2 supplies examples of transformations from PRML to the Hera methodology. Finally Section 8.3 presents the conclusions of the chapter.

#### 8.1 Transformations from PRML to the specifics of Hera

In this section the transformations from PRML to the specifics of the Hera approach are defined. Following the QVT standard, we have developed a set of relations to transform a PRML rule into a SeRQL [OpenRDF] expression that represents the same rule according to the Hera approach. This set of relations is explained along the next sections. To properly define these transformations we need to know both metamodels (PRML Lite and Hera). The PRML Lite metamodel was explained in Chapter 4. Next we give an overview of the Hera adaptation metamodel.

8.1.1 The Hera Adaptation metamodel

The Hera methodology [Houben et al, 2004; Vdovjak et al, 2003] is a model-driven approach for designing and developing Web Information Systems (WISs). As explained in Chapter 2 (see Section 2.3.5 for details) Hera defines an **Adaptation Model**, to specify how adaptation is carried out by using the data coming from UP or UM. This is performed by using inclusion conditions attached to model elements. Conditions based on UP attributes define the so-called *adaptability* - presentation is fixed before user browsing - while conditions based on UM model define the so-called *adaptivity* - presentation changes during user browsing - of the system. These appearance conditions over the different model elements are expressed using an extension of the SeRQL [OpenRDF] query language implemented by the HPG-Java engine. This engine is a Java servlet that, based on Hera models, generates pages on demand.

In Figure 8-1, an excerpt of the Hera metamodel for supporting adaptivity is shown. It is extended with the SeRQL syntax needed to specify the appearance conditions and queries on the different elements of the Application Model.

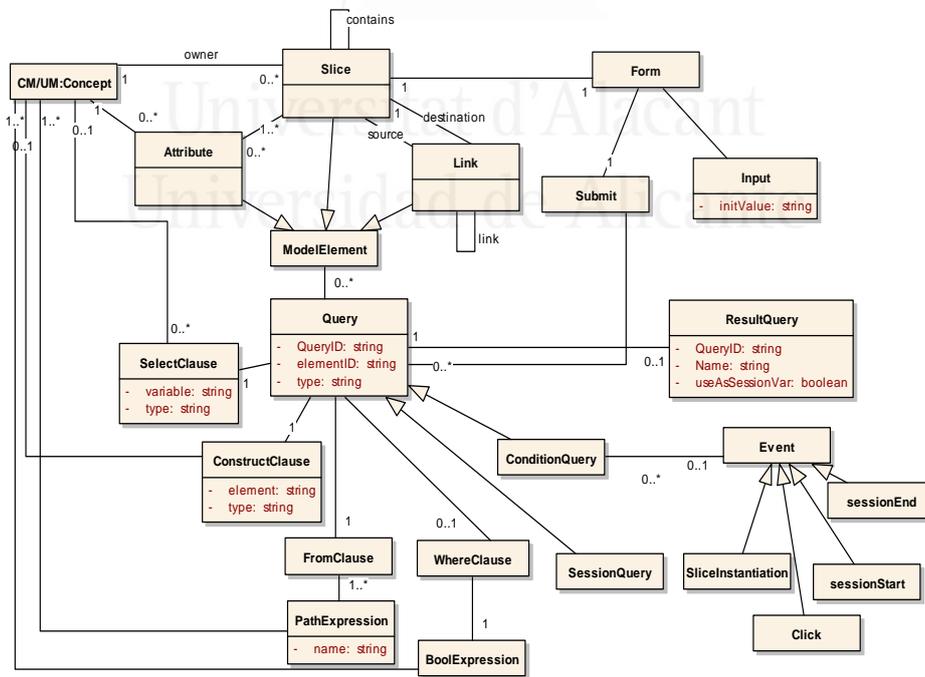


Figure 8-1: Excerpt of the Hera Metamodel (extended with SeRQL)

In next section, we will explain the set of QVT relations defined to transform PRML to the specifics of Hera.

### 8.1.2 Sequence of QVT Relations

In this section we present the QVT relations to transform a PRML rule into the specifics of the Hera approach. In order to apply the QVT relations, a model of each PRML rule must be defined according to the PRML Lite metamodel (presented in Chapter 4, Figure 4-3). A PRML rule is transformed into a set of SeRQL queries attached to an element of the Hera NM<sup>1</sup> (a Slice, Link or Attribute) representing the same personalization rule for the Hera approach. These queries allow data manipulations and some of them are conditions determining the set of instances of concepts, links or attributes to be displayed. The mentioned queries are based on an extension to the SeRQL query language and thus contain path expressions and conditions.

It is quite straightforward to obtain a model from PRML and SeRQL expressions, useful for applying the QVT transformations, and vice versa. Therefore, in this chapter we focus on presenting PRML and Hera queries in textual notation, although the QVT relations have been defined over their corresponding metamodels (PRML and SeRQL).

---

<sup>1</sup> The target of this mapping is the HPG-Java engine. This engine is a Java servlet that based on the Hera models (DM, NM and PM) generates pages on demand.

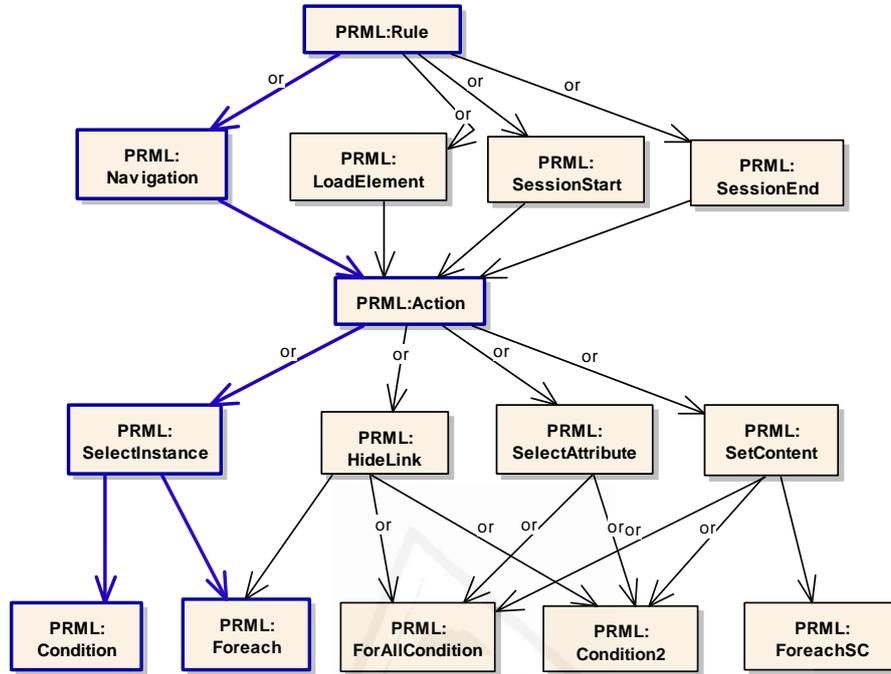


Figure 8-2 Sequence of Transformations

In Figure 8-2 we can see the sequence of the transformations that needs to be performed for the different types of PRML rules. This set of mappings is applied according to the structure of a PRML rule (see section 4.2 of Chapter 4): first the *PRMLRule* relation is applied to drive the whole transformation. In this way, this rule decides about which is the following mapping to be applied (according to the expression defined in the “where” clause). If the event is *Navigation* then the *PRMLNavigation* relation is applied. The *PRMLNavigation* relation has to call to the *PRMLAction* relation, since the main part of a PRML rule is the Action to be performed. The *PRMLAction* relation deals with executing the right relation according to the kind of Action. Then, if the Action is for example *SelectInstance*, the *PRMLSelectInstance* relation is applied. This relation together with *PRMLCondition* and *PRMLForeach* relations are used for creating the corresponding SeRQL model from the PRML model.

This set of QVT relations defined for transforming PRML to the specifics of the Hera approach are explained in detail along the next sections.

### 8.1.2.1 PRMLRule Relation

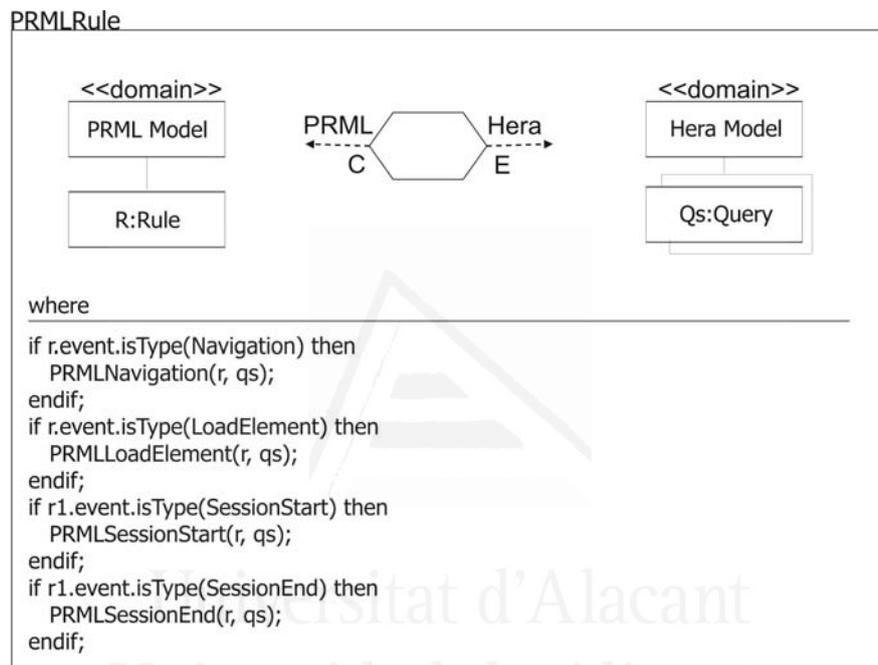


Figure 8-3 PRMLRule relation

A PRML rule is transformed into a Hera set of queries. The “where” clause checks the event type of the PRML rule calling the appropriate QVT relation.

### 8.1.2.2 Events Relations

There are four QVT relations defined for specifying the transformations from the different types of events in PRML Lite: *PRMLNavigation*, *PRMLLoadElement*, *PRMLSessionStart* and *PRMLSessionEnd*.

8.1.2.2.1 PRMLNavigationRelation

A PRML navigation event is transformed into a *Condition query* applied over the node activated by the event and into a *Session query* (with a *Result query* attached) for storing the parameter of the PRML navigation event as a Hera session variable (Figure 8-4). The condition query has attached a Hera *click* event. The “where” clause contains a relation that extends the previous one (i.e. PRMLAction) and it is described in Section 8.1.5. The Session query is different depending on the type of the parameter of the navigation event as we can see in Figures 8-4 and 8-5.

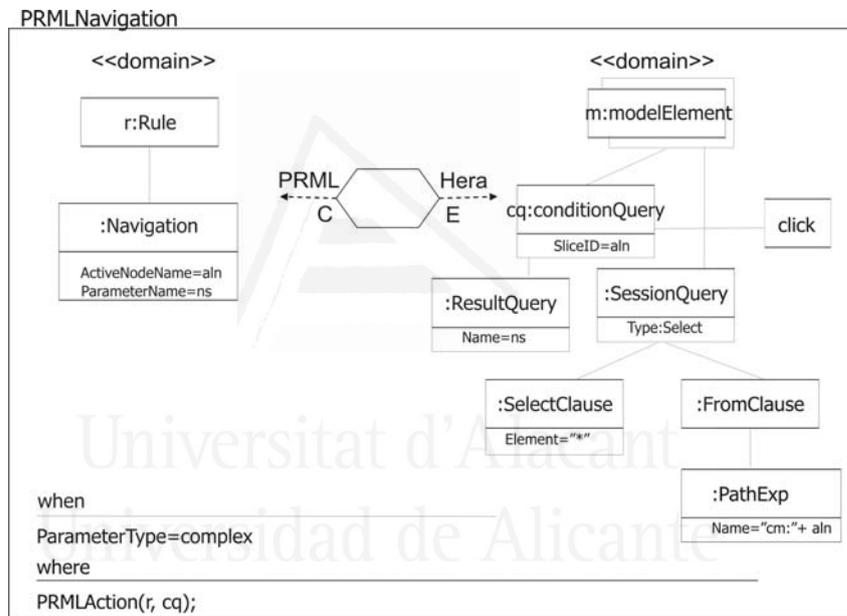


Figure 8-4 PRMLNavigation relation, complex parameter

The QVT relation is overloaded and depending on the type of the parameter (complex or simple) one of the relations is applied.

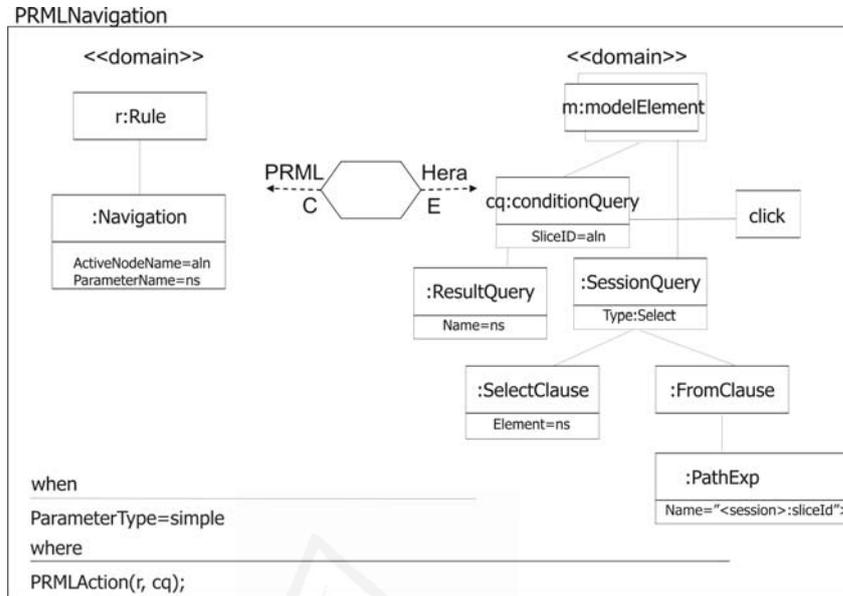


Figure 8-5 PRMLNavigation relation, simple parameter

### 8.1.2.2.2 PRMLLoadElement Relation

If the PRML rule is not triggered by a *navigation* event but by a *loadElement* event this relation is applied. A PRML *loadElement* event is transformed into a Hera *user browsing* event. The difference with the previous relation is that in this case the event is caused when a node is loaded (independently of the link which loaded it). The condition query has attached a Hera *SliceInstantiation* event. The “where” clause contains a relation (*PRMLAction*) that extends the previous one which is described in Section 8.1.5. In this case, the same as in the previous relation, the Session query is different depending on the type of the parameter of the loadElement event as we can see in Figures 8-6 and 8-7. This QVT relation is also overloaded and depending on the type of the parameter (complex or simple) one of the relations is applied.

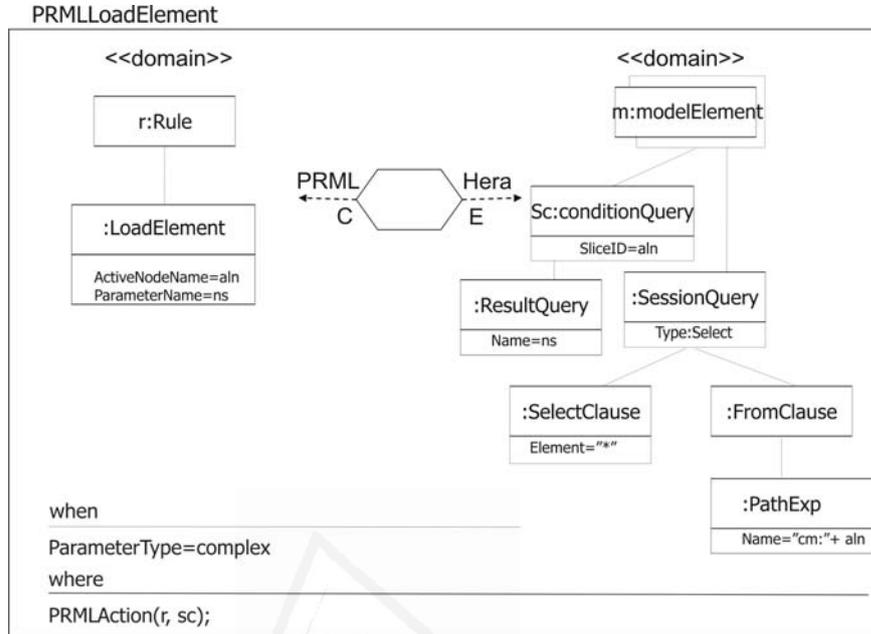


Figure 8-6 PRMLLoadElement relation, complex parameter

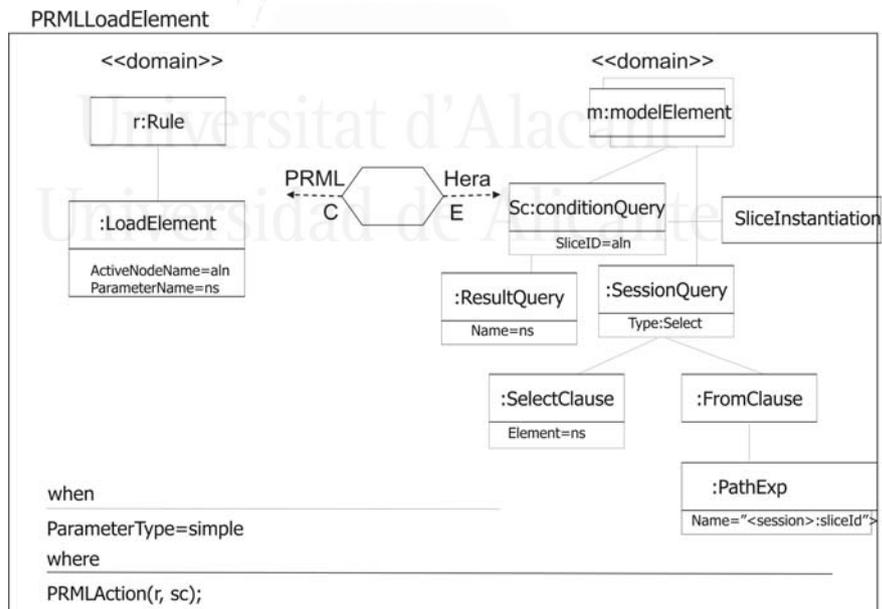


Figure 8-7 PRMLLoadElement relation, simple parameter

8.1.2.2.3 *PRMLSessionStart Relation*

If the PRML rule is triggered by a *SessionStart* event the *PRMLSessionStart* relation is applied. A PRML *SessionStart* event is transformed into a Hera *SessionStart* event. This transformation is straightforward. The “where” clause contains a relation (*PRMLAction*) that extends the previous one. The *PRMLAction* relation is described in Section 8.1.5.

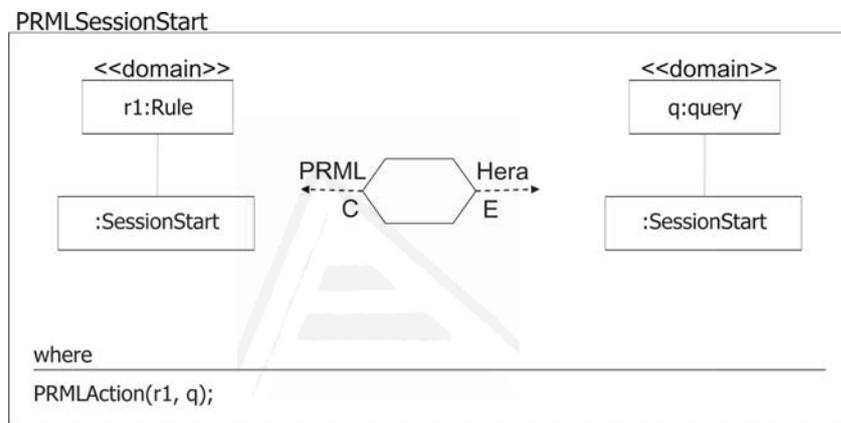


Figure 8-8 PRMLSessionStart relation

8.1.2.2.4 *PRMLSessionEnd Relation*

Finally, if the PRML rule is triggered by a *SessionEnd* event the *PRMLSessionEnd* relation is applied. A PRML *SessionEnd* event is transformed into a Hera *SessionEnd* event. The “where” clause contains a relation (*PRMLAction*) that extends the previous one, described in next section.

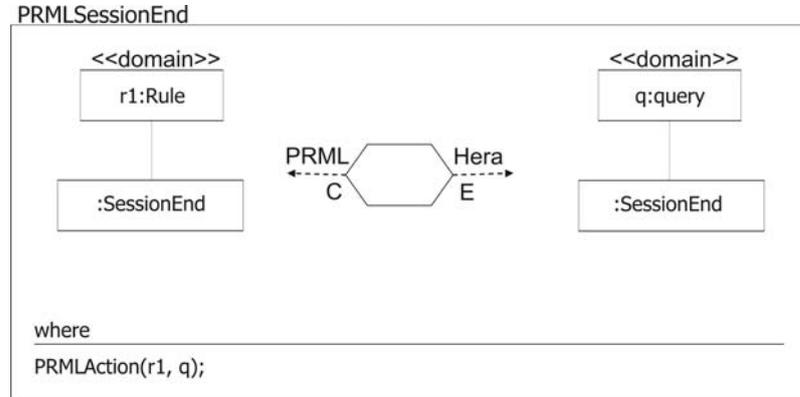


Figure 8-9 PRMLSessionEnd relation

### 8.1.2.3 PRMLAction Relation

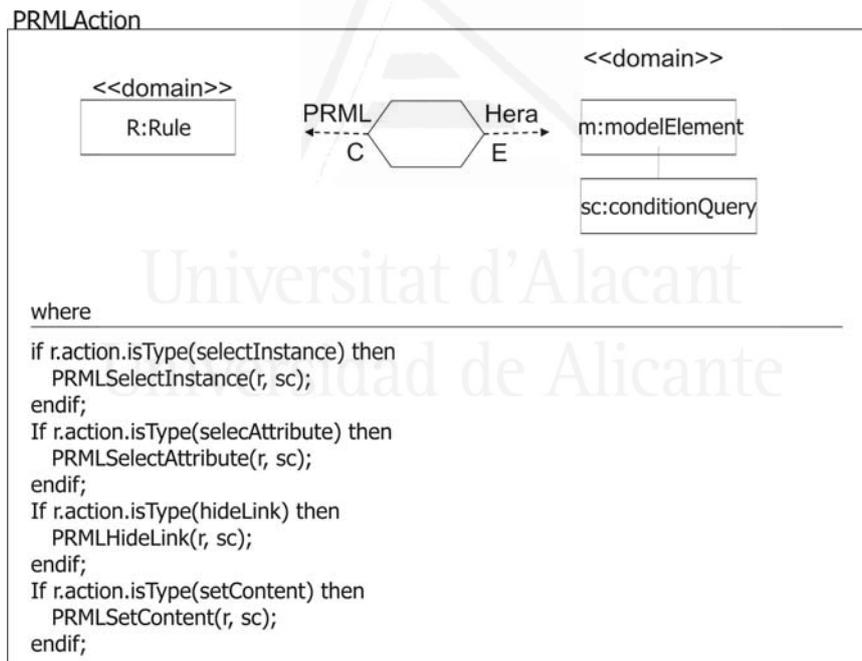


Figure 8-10 PRMLAction relation

After applying the *PRMLRule* relation and one of the events relations defined (depending on the event that triggers the PRML rule to be transformed), the

*PRMLAction* relation checks the kind of action in the PRML rule in order to call the adequate relation in the “where” clause.

#### 8.1.2.4 Actions Relations

A PRML action is transformed into a SeRQL query that represents the Hera rule. Depending on the type of action this query is built differently. There are four QVT relations defined for specifying the transformations from the different types of actions in PRML Lite: *PRMLSelectInstance*, *PRMLSelectAttribute*, *PRMLHideLink* and *PRMLSetcontent*.

##### 8.1.2.4.1 *PRMLSelectInstance* Relation

This relation checks that there is a set of elements in the PRML rule that represents a *SelectInstance* action according to the PRML Lite metamodel (see Fig. 4-3). These elements are: a *SelectInstance* class together with the corresponding Variable (that represents a collection) and the Type of this Variable.

This relation enforces that the corresponding Hera appearance condition is a query of type “Select” where SelectClause acts over a collection of elements of the same type of the Variable of the PRML SelectInstance (see Figure 8-11).

Once this relation holds, the *PRMLForEach* and the *PRMLCondition* relations must be performed, according to the “where” clause. In this way, every part of a PRML rule will be checked with its corresponding QVT rule, and the corresponding elements in the SeRQL will be created. These relations are explained in the Section 8.1.7.1 and 8.1.7.3 respectively.

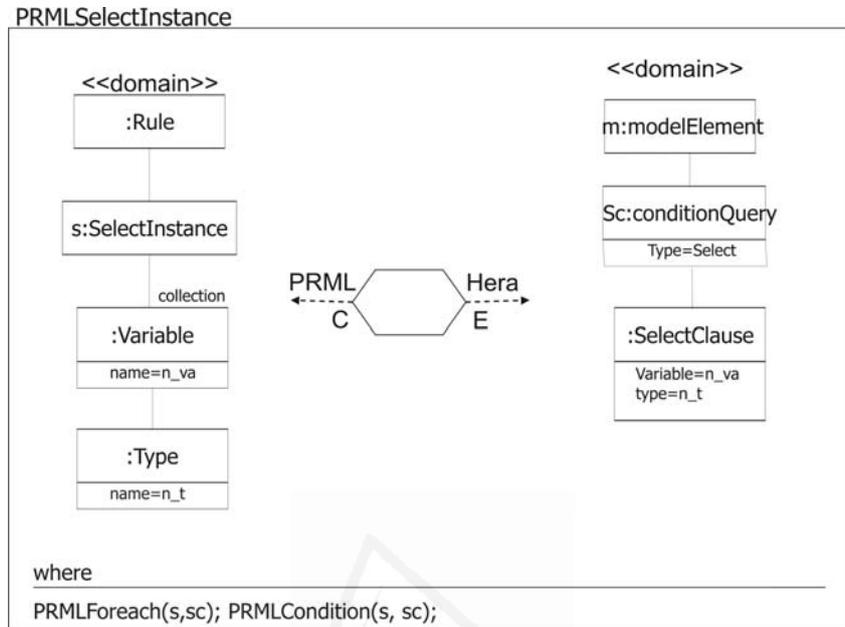


Figure 8-11 PRMLSelectInstance relation

#### 8.1.2.4.2 PRMLSelectAttribute Relation

This relation checks that there exists a set of elements in the PRML rule that represents a *SelectAttribute* action according to the PRML Lite metamodel (see Fig. 4-3). These elements are: a *SelectAttribute* class together with the corresponding *Variable* (that represents a collection) and the *Type* of this *Variable*, and a *ModelElement* which represents the attribute to be selected.

This relation enforces that the corresponding Hera appearance condition is a query of type “Select” where *SelectClause* is not relevant (because in Hera this action defined as an appearance condition which is applied over the corresponding attribute) and thus is a static expression (see Figure 8-12).

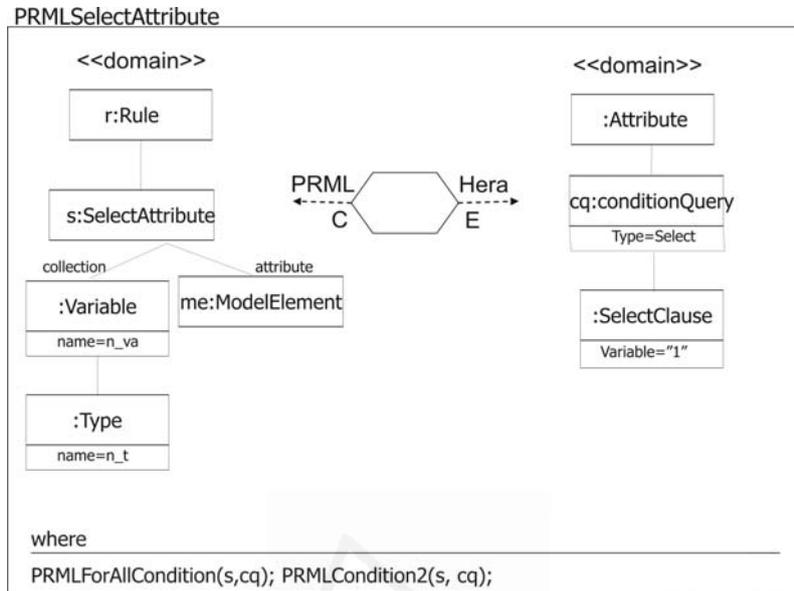


Figure 8-12 PRMLSelectAttribute relation

Once this relation holds, the *PRMLForAllCondition* and the *PRMLCondition2* relations must be performed, according to the “where” clause. In this way, every part of a PRML rule will be checked with its corresponding QVT rule, and the corresponding elements in the SeRQL will be created. These relations are explained in the Section 8.1.7.1 and 8.1.7.3 respectively.

#### 8.1.2.4.3 PRMLHideLink Relation

This relation checks that there is a set of elements in the PRML rule that represents a *HideLink* action according to the PRML Lite metamodel (see Fig. 4-3). These elements are: a *HideLink* class together with a *ModelElement* which represents the link to be hidden.

This relation enforces that the corresponding Hera appearance condition is a query of type “Select” where *SelectClause* is not relevant (because in Hera

this action defined as an appearance condition which is applied over the corresponding link) and thus is a static expression (see Figure 8-13).

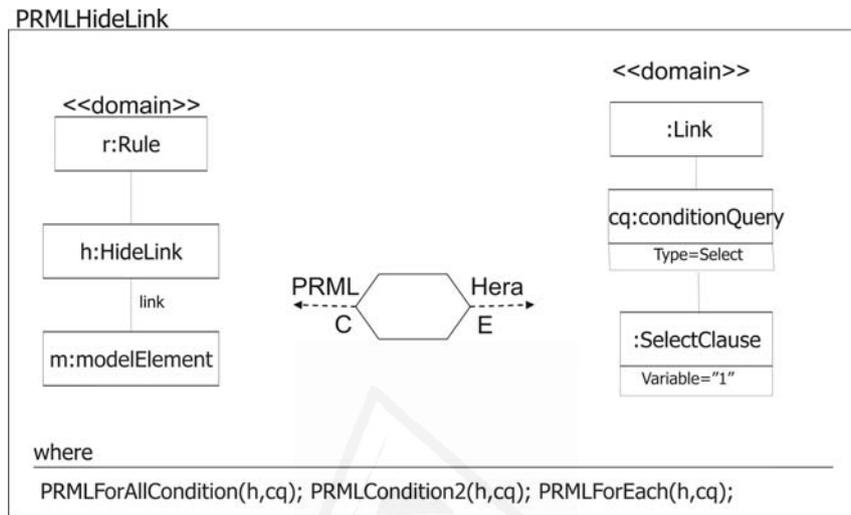


Figure 8-13 PRMLHideLink relation

Once this relation holds, the *PRMLForEach*, *PRMLForAllCondition* and the *PRMLCondition2* relations must be performed, according to the “where” clause. In this way, every part of a PRML rule will be checked with its corresponding QVT rule, and the corresponding elements in the SeRQL will be created. These relations are explained in the Section 8.1.7.1, 8.1.7.2 and 8.1.7.3 respectively.

#### 8.1.2.4.4 PRMLSetContent Relation

This relation checks that there is a set of elements in the PRML rule that represents a *SetContent* action according to the PRML Lite metamodel (see Fig. 4-3). These elements are: a *SetContent* class together with a *ModelElement* which represents the attribute to be updated.

This relation enforces three queries in Hera: a condition query which selects the element to be updated, a query which removes the old value of this

element, and a construct query which creates the element with the new value (see Figure 8-14). Note that in Hera we cannot update directly a value of an attribute, instead of this, we must delete the attribute and create it again with the new value.

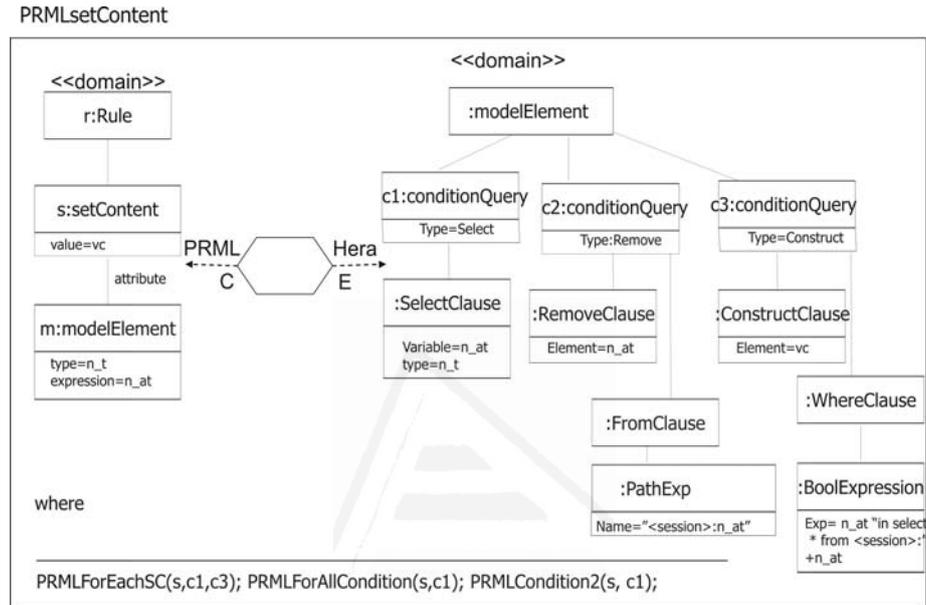


Figure 8-14 PRMLSetContent relation

Once this relation holds, the *PRMLForEachSC*, the *PRMLCondition2* and the *PRMLForAllCondition* relations must be performed, according to the “where” clause. Note that the delete query is completely created in this relation but the other two queries have to continue being built in next relations. These relations are explained in the Sections 8.1.7.2, 8.1.7.4 and 8.1.7.5 respectively.

### 8.1.2.5 Foreach and Condition Relations

There are different relations for mapping the PRML foreach and condition expressions. This is because depending the type of action performed it is mapped differently.

#### 8.1.2.5.1 PRMLForEach Relation

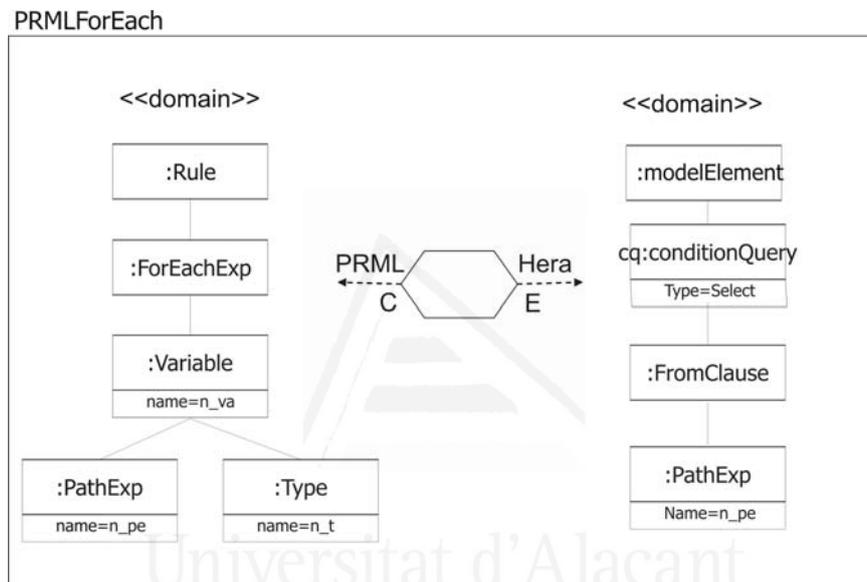


Figure 8-15 PRMLForEach relation

This relation checks that there is the following set of elements in the PRML rule: a *ForEach* element, with a *Variable* which has a *Type* and a *PathExpression*.

This relation enforces that the Hera Select query has a *from clause* with the same *PathExpressions* as in the PRML *ForEach expression* (Figure 8.15)

#### 8.1.2.5.2 PRMLForEachSC Relation

This relation checks that there is the following set of elements in the PRML rule: a *ForEach* element, with a *Variable* which has a *Type* and a *PathExpression*. This relation enforces that the select and construct queries have a from clause which path expression is the same as the path expression of the *ForEach Expression*.

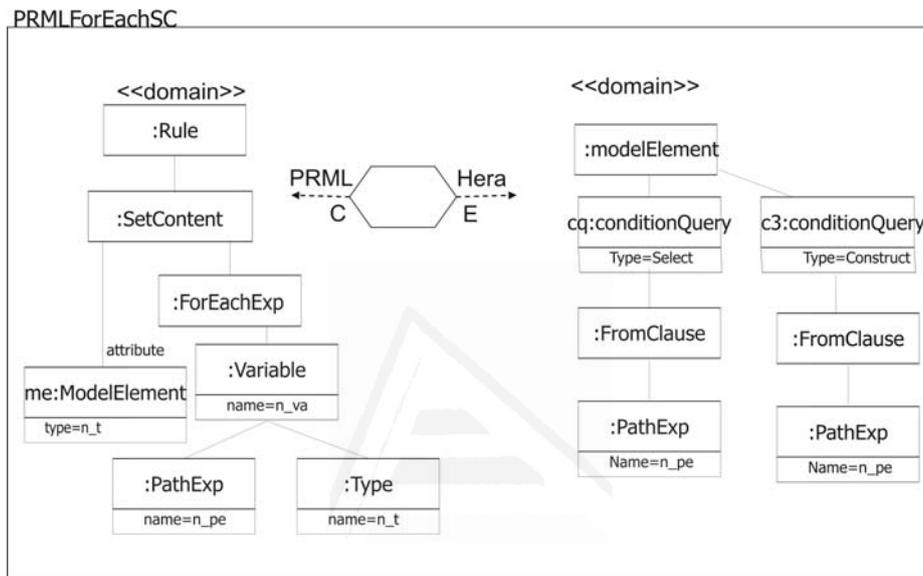


Figure 8-16 PRMLForEachSC relation

### 8.1.2.5.3 PRMLCondition Relation

This relation checks the *Condition* class together with its boolean expression (BoolExp) within the PRML rule.

In Hera the transformations of these elements enforce the *where clause* of the appearance condition as an expression that corresponds to the boolean expression of the condition element in the PRML rule (Figure 8.17).

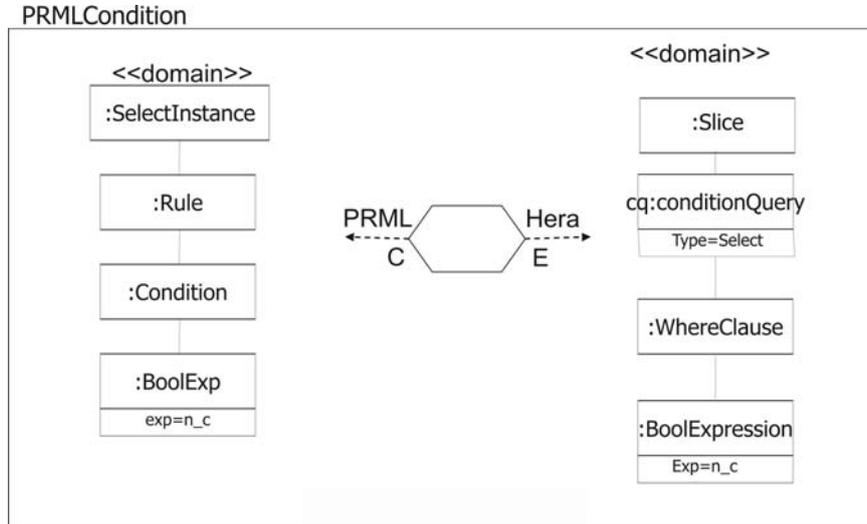


Figure 8-17 PRMLCondition relation

#### 8.1.2.5.4 PRMLCondition2 Relation

This relation checks the Condition class together with its boolean expression (BoolExp) within the PRML rule. In Hera the transformations of these elements enforce the *where clause* of the appearance condition as an expression that corresponds to the boolean expression of the condition element in the PRML rule. It also enforces that the *from clause* of the appearance condition is built with part of the boolean expression of the condition element in the PRML rule (Figure 8.18).

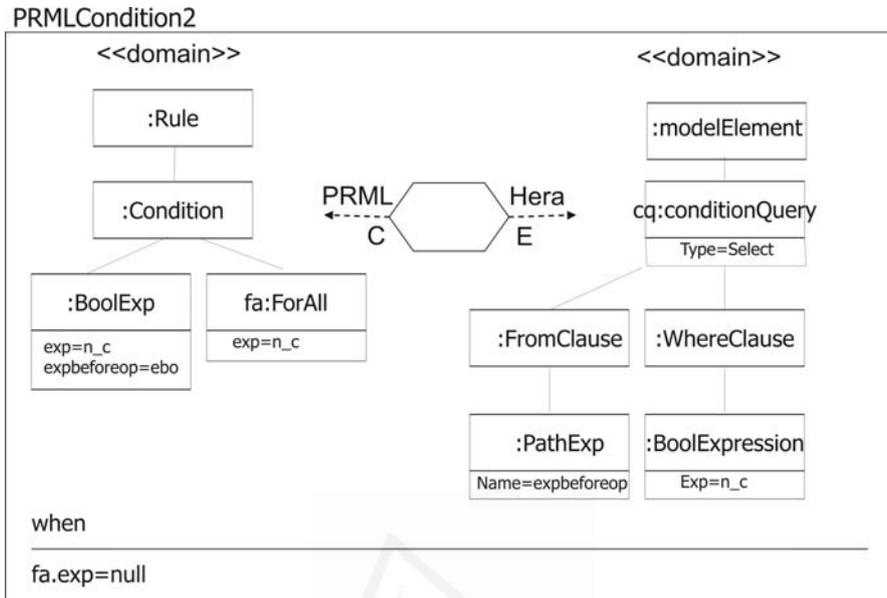


Figure 8-18 PRMLCondition2 relation

8.1.2.5.5 PRMLForAllCondition Relation

This relation checks the *Condition* class together with its boolean expression (*BoolExp*) and the *forall* statement within the PRML rule. In Hera the transformations of these elements enforce the *where clause* of the appearance condition as an expression with the keyword “ALL” followed by the boolean expression of the condition element in the PRML rule. It also enforces that the *from clause* of the appearance condition is built with part of the boolean expression of the condition element in the PRML rule (Figure 8.19).

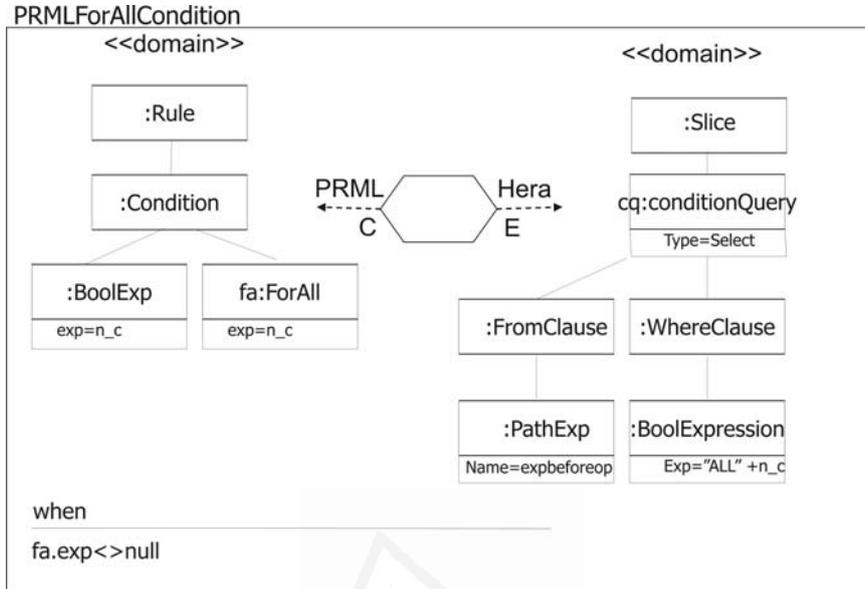


Figure 8-19 PRMLForAllCondition relation

### 8.1.3 Process Transformation Example

As an example, let's see how is performed the transformation needed for the following PRML rule. This rule was defined in section 4.3.3 of Chapter 4 to fulfil the 2<sup>nd</sup> personalization requirement described in the running example (i.e. offer recommendations based on the user interest on authors). This rule selects the instances (to be displayed) of the book concept which are written by an author in which the user has interest degree greater than 100.

```

When Navigation.Recommendations(NM.Book* books) do
  Foreach b,a in (books,UM.User.usertoInterest.interestToAuthor)
  do
    If (UM.User.userToInterest.degree > 100 and
    a.ID =b.booksToAuthor.ID) then
      books->selectInstance(b)
    endIf
  endForeach
endWhen

```

In Hera the corresponding set of queries is obtained by applying the QVT relations in the following way:

- **PRMLRule:** A PRML rule is transformed into a set of queries.
- **PRMLNavigation:** A PRML navigation event is transformed into a query condition over the recommendations slice. (*SliceCondition\_ID1*) and into a session and result queries (*Query\_ID1* and *QueryResult\_ID1*) storing the set of books received as an event parameter in the PRML rule as a session variable.
- **PRMLAction:** A PRML action is transformed into a condition query that has to be built depending on the action type.
- **PRML SelectInstance:** Once checked that the elements representing the *SelectInstance* action exist in the PRML rule, the relation enforces that the corresponding *Condition* query has a *SelectClause* in the appearance condition acting over a collection of elements of the same type of the variable of the PRML *SelectInstance*.

```
SELECT B
```

- **PRMLForEach:** The ForEach element (with a variable which has a type and a PathExp) of the PRML rule is transformed into a *from clause* in the *Condition query* with the same *PathExpressions* as in the PRML *foreach expression*.

```
FROM
```

```
{B}<cm:booksToAuthor>(BTA)<um:ID>{ID},
<um:userToInterest>{UI}<um:degree>{D},
<um:userToInterest>{UTI}<um:interestToAuthor>{ITA}<um:ID>{ID2},
{session}<session:user>{U}
```

- **PRMLCondition:** The body of the IteratorExp is defined in the *where clause* of the query *select* of the appearance condition as an expression that corresponds to the boolean expression of the condition element in the PRML rule.

```
WHERE D>100 and ID= ID2 and B in Select * from session:books
```

---

The set of queries resulting from the transformation are the following:

### Session and Result Queries

This set of queries fills the “book” session variable with the set of books of the domain model

```
<rdfs:Class rdf:ID="Query_ID1">
<rdfs:subClassOf rdf:resource="&Query"/>
  <slice:queryString>
    SELECT * FROM <cm:books>
  </slice:queryString>
</rdfs:Class>

<rdfs:Class rdf:ID="QueryResult_ID1" slice:resultName=books
slice:useAsSessionVar="Yes">
  <rdfs:subClassOf rdf:resource="&slice:QueryResult"/>
</rdfs:Class>

<rdf:Property rdf:ID="result-ref_ID1">
  <rdfs:subPropertyOf rdf:resource="&Slice#result-ref"/>
  <rdfs:domain rdf:resource="#Query_ID1" />
  <rdfs:range rdf:resource="#QueryResult_ID1" />
</rdf:Property>
```

---

### Condition Query

This query is a condition over the recommendation slice. It checks the conditions specified in the 2<sup>nd</sup> requirement (i.e. the interest degree in a book’s author has to be greater than 10). If the condition holds, the proper instances of the recommendation slice will be visible

```
<rdfs:Class rdf:ID:"SliceCondition_ID1">
<rdfs:subclassOf
rdf:resource="http://wwwis.win.tue.nl/~hera/ns/slice#SliceCondition"/>
  <slice:queryString>
    SELECT B
    FROM
    {B}<cm:booksToAuthor>(BTA)<um:ID>{ID},
    <um:userToInterest>{UI}<um:degree>{D},
    <um:userToInterest>{UTI}<um:interestToAuthor>{ITA}<um:ID>{ID2},
    {session}<session:user>{U}

    WHERE D>100 and ID= ID2 and B in Select * from session:books
  </slice:queryString>
</rdfs:Class>

<rdf:Property rdf:ID="condition-ref_ID1">
  <rdfs:subPropertyOf rdf:resource="&Slice#condition-ref"/>
  <rdfs:domain rdf:resource="#SLice.Recommendation_ID1" />
  <rdfs:range rdf:resource="#SliceCondition_ID1" />
```

</rdf:Property>

Next, examples of transformations using the different types of actions are shown.

### 8.2 PRML To Hera: Transformation Examples

The examples shown in this section refer to the different actions of PRML, which are based in the running example described in Chapter 4 (i.e. an online library). The corresponding Hera models (i.e. domain and user model and navigation model) are shown next. Note that is not the purpose to explain in detail these models but just help in the understanding of the examples.

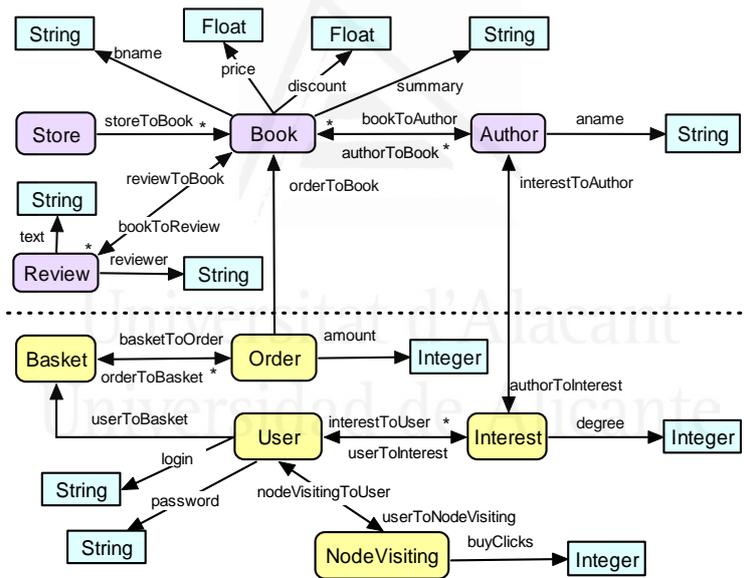


Figure 8.20 DM-UM for Hera

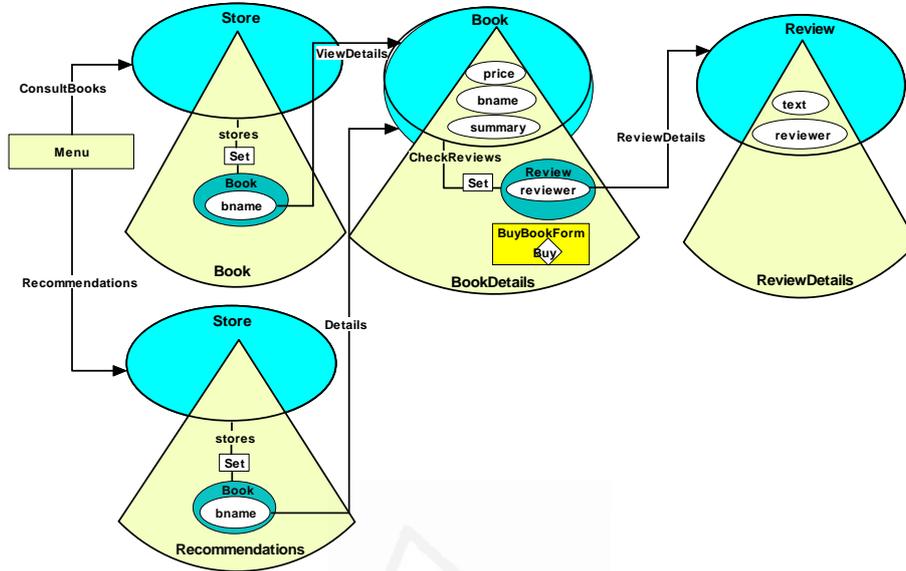


Figure 8.21. Hera Navigation Model

Figure 8.21 shows the NM for the Hera approach for the case study of the dissertation presented in Chapter 4. We can see that this model has the same expressiveness as the UWE NM presented in the previous chapter, and the A-OOH NM presented in Chapter 4. In the Hera approach the nodes are represented by slices. For more details on the Hera models consult [Houben et al, 2004].

To fulfill the personalization requirement described in previous section (users will see recommendations of books based on their interests in authors) we need to store the interest of the user in the User Model as described before. We consider that the user is interested in an author when he consults the details of a book of this author. The PRML rule updating the interest of the user in the User Model is explained in section 4.3.1, its counterpart in Hera is next:

- *SetContent Example*

	<code>When Navigation.ViewDetails (NM.Book b) do</code>
--	---

PRML	<pre> <b>Foreach</b> i <b>in</b> (UM.User.userToInterest) <b>do</b>   <b>If</b>(b.bookToAuthor.ID = i.interestToAuthor.ID) <b>then</b>     <b>setContent</b>(i.degree,i.degree+10)   <b>endIf</b> <b>endForeach</b> <b>endWhen</b> </pre>
Hera	<pre> A) &lt;rdfs:Class rdf:ID:"<b>Query_ID0</b>"&gt; &lt;rdfs:subClassOf rdf:resource="&amp;Query"/&gt;   &lt;slice:queryString&gt;     <b>select</b> B     <b>from</b> {session}&lt;session:sliceid&gt;{B}   &lt;/slice:queryString&gt; &lt;/rdfs:Class&gt;  &lt;rdfs:Class rdf:ID:"<b>QueryResult_ID0</b>" slice:resultName="<b>b</b>" slice:<b>userAsSessionVar</b>="<b>Yes</b>"&gt; &lt;rdfs:subClassOf rdf:resource="&amp;slice:QueryResult"/&gt; &lt;/rdfs:Class&gt;  &lt;rdf:Property rdf:ID:"result-ref_ID0"&gt;   &lt;rdfs:subPropertyOf rdf:resource="&amp;Slice#result- ref"/&gt;   &lt;rdfs:domain rdf:resource="##<b>Query_ID0</b>" /&gt;   &lt;rdfs:range rdf:resource="##<b>QueryResult_ID0</b>" /&gt; &lt;/rdf:Property&gt; ----- B) &lt;rdfs:Class rdf:ID:"<b>Query_ID1</b>"&gt; &lt;rdfs:subPropertyOf rdf:resource="&amp;Query"/&gt; &lt;slice:queryString&gt;   <b>select</b> D   <b>from</b> {session}&lt;session:user&gt;{U}, {U}&lt;UM:userToInterest&gt;{I}&lt;UM:degree&gt;{D}, {I}&lt;UM:interestToAuthor&gt;{A}, {B}&lt;CM:bookToAuthor&gt;{A1}, {session}&lt;session:b&gt;{B1}    <b>where</b> A = A1 and B=B1 &lt;/slice:queryString&gt; &lt;/rdfs:Class&gt;  &lt;rdfs:Class rdf:ID:"<b>QueryResult_ID1</b>" slice=resultName:"<b>degree</b>" slice:<b>userAsSessionVar</b>="Yes"&gt; &lt;rdfs:subClassOf rdf:resource="&amp;slice:QueryResult"/&gt; &lt;/rdfs:Class&gt; </pre>

	<pre> &lt;rdf:Property rdf:ID:"result-ref_ID1"&gt;   &lt;rdfs:subPropertyOf rdf:resource="&amp;Slice#result- ref"/&gt;   &lt;rdfs:domain rdf:resource:"#Query_ID1"/&gt;   &lt;rdfs:range rdf:resource="&amp;QueryResult_ID1"/&gt; &lt;/rdfs:Property&gt;  -----  C) &lt;rdfs:Class rdf:ID:"Query_ID2"&gt; &lt;rdfs:subPropertyOf rdf:resource="&amp;Query"/&gt; &lt;slice:queryString&gt;   Remove D   From {session}&lt;session:user&gt;{U},       {U}&lt;UM:userTointerest&gt;{I}&lt;UM:degree&gt;{D} &lt;/slice:queryString&gt; &lt;/rdfs:Class&gt;  &lt;rdfs:Class rdf:ID:"Query_ID3"&gt; &lt;rdfs:subPropertyOf rdf:resource="&amp;Query"/&gt; &lt;slice:queryString&gt;   Construct distinct {I}&lt;um:degree&gt;{D+10}   From {U}&lt;UM:userTointerest&gt;{I},       {session}&lt;session:user&gt;{U}   Where D in select * from session:degree &lt;/slice:queryString&gt; &lt;/rdfs:Class&gt;  &lt;rdf:Property rdf:ID:"query_ID2"&gt;   &lt;rdfs:subPropertyOf rdf:resource="&amp;Slice#query"/&gt;   &lt;rdfs:domain rdf:resource:"#Slice.Book"/&gt;   &lt;rdfs:range rdf:resource="&amp;Query_ID2"/&gt;   &lt;slice:order&gt;1&lt;/slice:order&gt; &lt;/rdf:Property&gt;  &lt;rdf:Property rdf:ID:"query_ID3"&gt;   &lt;rdfs:subPropertyOf rdf:resource="&amp;Slice#query"/&gt;   &lt;rdfs:domain rdf:resource:"#Slice.Book"/&gt;   &lt;rdfs:range rdf:resource="&amp;Query_ID3"/&gt;   &lt;slice:order&gt;2&lt;/slice:order&gt; &lt;/rdf:Property&gt; </pre>
--	--

We can divide the set of Hera queries into three parts:

- a) This part fills the selected book to the “book” variable. It is fired when a slice describing the particular book is visited.

- b) This part reads the interest degree of the current user in the particular book and assigns it to “interest” variable.
- c) This part updates the value – 1st query removes the old one and the second creates a new value.

- *SelectAttribute Example*

To illustrate the SelectAttribute action of PRML Lite, we consider the 1<sup>st</sup>: personalization requirement defined in the case study: When the user clicks on the *ViewDetails* link to consult a specific book, if the number of books bought is greater than 10, we show an attribute “discount” to the user (this rule was explained in the Section 4.3.2 of Chapter 4).

<b>PRML</b>	<pre> When Navigation.ViewDetails(NM.BookDetails book) do If UM.User.uToNodeVisiting.buyClicks &gt;10 then     book.Attributes.selectAttribute(discount) endif endWhen </pre>
<b>Hera</b>	<pre> &lt;rdfs:Class rdf:ID="Query_ID1"&gt; &lt;rdfs:subClassOf rdf:resource="&amp;Query"/&gt; &lt;slice:queryString&gt;     SELECT * FROM cm:books &lt;/slice:queryString&gt; &lt;/rdfs:Class&gt;  &lt;rdfs:Class rdf:ID="QueryResult_ID1" slice:resultName=books slice:useAsSessionVar="Yes"&gt; &lt;rdfs:subClassOf rdf:resource="&amp;slice:QueryResult"/&gt; &lt;/rdfs:Class&gt;  &lt;rdf:Property rdf:ID="result-ref_ID1"&gt; &lt;rdfs:subPropertyOf rdf:resource="&amp;Slice#result-ref"/&gt; &lt;rdfs:domain rdf:resource="#Query_ID1" /&gt; &lt;rdfs:range rdf:resource="#QueryResult_ID1" /&gt; &lt;/rdf:Property&gt; ----- - &lt;rdfs:Class rdf:ID:"SliceCondition_ID1"&gt; &lt;rdfs:subclassOf rdf:resource="http://wwwis.win.tue.nl/~hera/ns/slice#SliceCondit ion"/&gt; </pre>

	<pre> &lt;slice:queryString&gt;   SELECT 1   FROM   {um:User}cm:userToNodeVisiting{UNV}cm:buyClicks{BC};   WHERE BC&gt;10 and B in Select * from session:books &lt;/slice:queryString&gt; &lt;rdfs:Class&gt;  &lt;rdf:Property rdf:ID="condition-ref_ID1"&gt;   &lt;rdfs:subPropertyOf rdf:resource="#Slice#condition- ref"/&gt;   &lt;rdfs:domain rdf:resource="#Attribute.Discount_ID1" /&gt;   &lt;rdfs:range rdf:resource="#SliceCondition_ID1" /&gt; &lt;/rdf:Property&gt; </pre>
--	--

This kind of rule is transformed in Hera into an appearance condition attached to the attribute “discount”. The parameter of the navigation event is stored into a session variable.

- **HideLink Example**

To illustrate the HideLink action of PRML Lite, we consider the 3rd requirement of the case study of this dissertation: If the user interest degree on the different authors is not initialized (this means the user has not still consulted the details of any book, maybe is the first time the user goes into the website) the *recommendations* link is not shown. Note that there is only one *Recommendations* link in the website. The PRML rule was explained in the Section 4.3.4 of the Chapter 4. The Hera counterpart is next:

<b>PRML</b>	<pre> When SessionStart do   If ForAll(UM.User.usertoInterest)     (UM.User.usertoInterest.degree=null or UM.User.usertoInterest.degree&lt;100) then     hideLink(NM.Recommendations)   endIf endWhen </pre>
	<pre> &lt;rdfs:Class rdf:ID:"SliceCondition_ID1"&gt; &lt;rdfs:subclassOf rdf:resource="http://wwwis.win.tue.nl/~hera/ns/slice#SliceCondit </pre>

<b>Hera</b>	<pre> ion"/&gt;     &lt;slice:queryString&gt;       SELECT 1       FROM       {um:User}cm:userToInterest{UTI}cm:degree{D};       WHERE ALL(D=null or D&lt;100)     &lt;/slice:queryString&gt;   &lt;rdfs:Class&gt;    &lt;rdf:Property rdf:ID="condition-ref_ID1"&gt;     &lt;rdfs:subPropertyOf rdf:resource="#&amp;Slice#condition- ref"/&gt;     &lt;rdfs:domain rdf:resource="#Link.Recommendations" /&gt;     &lt;rdfs:range rdf:resource="#SliceCondition_ID1" /&gt;   &lt;/rdf:Property&gt; </pre>
-------------	---

This kind of rule is transformed in Hera into an appearance condition attached to the link “Recommendations”.

### 8.3 Conclusions

In this chapter it has been explained how PRML Lite can be exported to the specifics of the Hera approach defining a set of QVT relations to conceptually show it. An example for each of the PRML actions considered is given for a better understanding of the transformations.

The Appendix C of this dissertation describes the *PRML translator*, a prototype tool that automatically implements the defined transformations.



Universitat d'Alacant  
Universidad de Alicante

## Chapter 9

### Conclusions

This final chapter summarizes the work presented in this dissertation. Moreover it stresses the main contributions (Section 9.1) and limitations (Section 9.2) of the present approach. Finally Section 9.3 presents the ongoing work and the possible future extensions.

#### 9.1 Contributions

The first aim of this dissertation was to add personalization to the Object Oriented Hypermedia (OO-H) approach. For this purpose, OO-H became A-OOH (adaptive OO-H), which design process is based on the Unified Process (UP) and not in the spiral model like OO-H. The reason for changing the design process is that the adaptive hypermedia systems need an appropriate software engineering process for their development. The design models of OO-H were also modified. First of all, to make them standard UML profiles were defined. Then the models were improved for the support of adaptive modeling and some models were added, concretely a presentation model and a user and personalization model. Chapter 3 presents the Adaptive OO-H (A-OOH) fundamentals describing its design process and the different diagrams considered. As a result of performing the activities of each of the design process phases, in A-OOH we get a set of models, reflecting views of the interface to be generated.

Web site modelling methodologies provide a conceptual and systematic approach to (complex) Web application design and implementation. Most of these methodologies allow developing Personalized Web sites where the personalization specifications are embedded in their design models. This can cause some problems like the difficulty of maintenance of the personalization and the inability of reusing the personalization specification among different approaches. In Chapter 4 we have presented a solution based on a high-level

ECA rule language called PRML (Personalization Rules Modeling Language) which allows to specify the personalization at design time as an orthogonal concern of the Web site, independent of the underlying technology. This language can be used by the designer to specify at design time the personalization to be performed at runtime (Chapters 4 and 5).

Moreover the fact of considering Personalization as a full discipline present in many fields besides Web Engineering asks for the role of a *designer* who is responsible of defining the personalization and who has knowledge of psychology and sociology. In consequence, the *personalization designer* does not have to be an experienced programmer. The *personalization designer* finds the problem of having as many ways of specifying the personalization as the number of existing Web methodologies which support it. Depending on the Web methodology used, the *personalization designer* should learn a different language for specifying personalization. This can be a cumbersome task for the designer, who, as aforementioned, does not have to be an experienced programmer. We argue that the designer can use PRML for specifying the personalization and then this specification is transformed to the specifics of the Web design method used (such transformations are explained in the Chapters 7 and 8). This would reduce the learning curve of the personalization designer.

The double purpose of PRML is making possible the reuse of personalization strategies among different approaches and allowing the definition of more complex (and specific) personalization actions. For this purpose, two conformance levels are defined in this language: PRML Lite and PRML Full.

PRML Full (explained in Chapter 5) provides the designer a way of specifying more complex personalization actions than PRML Lite (presented in Chapter 4) limiting the reusability. PRML Full actions are needed for specifying more advanced personalization strategies. For this purpose it also allows recognizing complex navigation behaviours of the user, needed for defining efficient personalization strategies.

Another important contribution is AWAC, a prototype CAWE tool for the automatic generation of adaptive Web applications using PRML (presented in Chapter 6). This tool implements the A-OOH (Adaptive Object Oriented

Hypermedia) methodology. The input of the AWAC tool is the set of A-OOH design models needed to model the adaptive Website to generate. The output is the generated Website with its database. Once generated, the adaptive Website also contains three modules for managing the personalization which, at runtime, analyze the user browsing events and adapt the Website according to the personalization rule(s) triggered. The personalization rules can be edited in a way that is independent of the rest of the application, which improves the personalization maintenance.

In this dissertation we argue that PRML Lite can be exported to the specifics of different Web methodologies. This implies that the PRML language is reusable as well as the personalization strategies defined with it. Chapters 7 and 8 explained how PRML Lite can be exported to the specifics of the UWE and Hera approaches defining a set of transformations to conceptually show it. These transformations are defined using the QVT language. The Appendix C of this dissertation describes the PRML translator, a prototype tool that automatically implements the defined transformations.

## 9.2 Limitations

Considering that is impossible to cover all the aspects of personalization some boundaries were set to this dissertation and some limitations exist:

- *Adapting the presentation.* We do not consider the personalization of the presentation in this work.
- *Limitation considering Context-Aware websites.* This work does not consider in detail the context based adaptation, as it is out of the scope of this work. However, the approach supports some kind of context-awareness. For this purpose the UM should be extended with context entities.
- *PRML FULL partially implemented.* All the PRML functionality has not been implemented in the AWAC tool. PRML Lite is completely implemented but we did not implement yet all the actions of PRML Full: grouping users, adding and deleting links, support of complex events and the behavioural rules.

- *AWAC tool: no graphical editor.* The AWAC tool does not have a graphical editor for drawing the A-OOH models.
- *Limitations of the portability of PRML.* Defining transformations from PRML to other approaches (besides UWE and Hera) would strengthen the obtained results. However it should be noted that being able to transform PRML to these different approaches is enough to show our claims.
- *Limitations of the experiment.* The number of visitors of the generated website is rather small, due to the subject of the case study. It would be interesting to make a larger website with a bigger number of visitors.

### 9.3 Future Work

- As future work, besides implementing all the PRML functionality we would like to add a graphical user interface in order to define the A-OOH models using AWAC. Now, to define the A-OOH models and generate the XMI files we use the Enterprise Architect Design tool [Enterprise Architect] in which we have defined the UML profiles needed for the modelling of the A-OOH diagrams.
- Another future action is to extend PRML Full to support adaptation actions over the presentation of the Website.
- We also plan to define transformations from PRML to other Web design approaches like WebML or OOHDm to improve the reusability of the language.
- Another task to perform is to model larger experiments to corroborate the results obtained in this dissertation. We would also like to explicitly evaluate the satisfaction of the users and the usability of the language by means of questionnaires and usability tests to verify the results concluded in this work.
- Moreover, we have the purpose of investigating eventual extensions of PRML full actions to better cover all possible personalization strategies and scenarios.
- We are also aimed at checking the consistency of the models before transformations. Therefore, before processing conditions and operations

we will check the parameters for their consistency with the navigation model structure and with the User Model.

- Studying the impact of the Web 2.0 and AJAX on PRML is also interesting to take into account for further experiments. For this purpose we plan to extend the set of PRML events, taking into account these issues.
- Regarding the AWAC tool we are currently working on an automatic installer. It actually works on Windows XP systems. We are still testing the installer under Windows Vista systems.



Universitat d'Alacant  
Universidad de Alicante

## Appendix A

### BNF specification of PRML Lite

#### *PRML Specification*

```

<PRMLversion> ::= 'PRMLversion:' <version> 'PersonalizationFrequency:'
<frequency>

<version> ::= 'Lite' | 'Full'

<frequency> ::= <nonzero-digit> | 'n'

<rules> ::= <rule> | <rule> <rules>

```

#### *Rule Specification*

```

<rule> ::= <features> 'When' <event> 'do' <body> 'endWhen'

<features> ::= 'Rule:' <ruleName> 'type:' <type> 'priority:' <priority>
'activation:' <activationDate> 'expiration:' <expirationDate>
'pfrequency:' <pfrequency>

<type> ::= ('Acquisition' | 'Personalization' | 'Mixed' )

<priority> ::= ('high' | 'medium' | 'low' )

<body> ::= (<foreachExp>)? ('If (' <condition>') then')? <action> 'endIf'?
'endForeach'?

```

#### *Event*

```

<event> ::= ('SessionStart' | <navigation> | <loadElement> | 'SessionEnd')

<navigation> ::= 'Navigation.' <activeLinkName> (<parameterExpression>)

<loadElement> ::= 'LoadElement.' <activeNodeName> (<parameterExpression>)

<parameterExpression> ::= 'NM.' <activeNodeName> '*'? <parameterName>

```

#### *Condition Expressions*

```

<condition> ::= (<boolExpression>) (<booloperator> (<boolExpression> )*)

<boolExpression> ::= <operand> <compOperator>
(<value> | <operand> | <literal>) | 'not' <boolExpression>

```

```

<foreachExp> ::= 'Foreach' (<variableName> ',') + 'in' ( ( <PE> |
<conceptName> ) , ) + 'do'

<operand> ::= <PE> ( ( <arithmeticOperator> ) ( <PE> | <number> ) ) ?

<PE> ::= [ 'UM.User' | 'DM.<conceptName> | <parameterName> ] { '.'
'<targetRole>' } * '.'
[ <attributeName> | <conceptName> ]

<PE> ::= 'NM.<nodeName>' { '<linkName>' } * '.' ( <attributeName> |
<nodeName> ) ? | <parameterName> { '.' <attributeName> }

<compOperator> ::= '<' | '>' | '=' | '<=' | '>=' | '<>'

<booloperator> ::= 'and' | 'or' | 'not'

<arithmeticOperator> ::= '+' | '-' | '*' | '/'

```

### Action

```

<action> ::= <setContent> | <selectInstance> | <selectAttribute> |
<hideLink>

<setContent> ::= 'setContent(' ( <PE> | <operand> ) ', ' ( <PE> | <operand> |
<literal> ) ? ')'

<selectInstance> ::= 'selectInstance(' <PE> | <parameterName> ', '
<variableName> ')'

<selectAttribute> ::= 'selectAttribute(' <PE> ')'

<hideLink> ::= 'hideLink(NM.<linkName>)'

```

### Miscellaneous

```

<ruleName> ::= <string>

<pfrequency> ::= <string>

<activeLinkName> ::= <string>

<activeNodeName> ::= <string>

<parameterName> ::= <string>

<variableName> ::= <string>

<conceptName> ::= <string>

<targetRole> ::= <string>

<attributeName> ::= <string>

<linkName> ::= <string>

```

```

<nodeName> ::= <string>
<value> ::= <number>+ | <boolean>

```

### Data Types

```

<number> ::= '-'? ( <integer> | <float> | <exponential> )
<integer> ::= <nonzero-digit> <digit> *
<float> ::= <integer> '.' <digit> <digit> *
<exponential> ::= <digit> '.' <digit> <digit> * 'E' ('+' | '-' ) <integer>
<digit> ::= '0' | <nonzero-digit>
<nonzero-digit> ::= '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
<boolean> ::= 'true' | 'false'
<string> ::= <char>+
<literal> ::= '"' ( <string> | ' ' | <number> | ',' | '%' ) * '"'
<char> ::= 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' | 'j' | 'k' |
'l' | 'm' | 'n' | 'o' | 'p' | 'q' | 'r' | 's' | 't' | 'u' | 'v' | 'w' | 'x' | 'y'
| 'z' | 'A' | 'B' | 'C' | 'D' | 'E' | 'F' | 'G' | 'H' | 'I' | 'J' | 'K' | 'L'
| 'M' | 'N' | 'O' | 'P' | 'Q' | 'R' | 'S' | 'T' | 'U' | 'V' | 'W' | 'X' | 'Y' |
'Z'

```



Universitat d'Alacant  
Universidad de Alicante

## Appendix B

### BNF specification of PRML Full

#### *PRML Specification*

```

<PRMLversion> ::= 'PRMLversion:' <version> 'PersonalizationFrequency:'
<frequency>

<version> ::= 'Lite' | 'Full'

<frequency> ::= <nonzero-digit> | 'n'

<rules> ::= <rule> | <rule> <rules>

```

#### *Rule Specification*

```

<rule> ::= <features> 'When' <event> 'do' <body> 'endWhen'

<features> ::= 'Rule:' <ruleName> 'type:' <type> 'priority:' <priority>
'activation:' <activationDate> 'expiration:' <expirationDate>
'pfrequency:' <pfrequency>

<type> ::= ('Acquisition' | 'Personalization' | 'Mixed' | 'Behaviour')

<priority> ::= ('high' | 'medium' | 'low' )

<body> ::= (<foreachExp>)? ('If (' <condition>') then')? <action> ['else'
<action>] 'endIf'? 'endForeach'?

```

#### *Event*

```

<event> ::= ('SessionStart' | <navigation> | <loadElement> | 'SessionEnd' |
'Init')

<navigation> ::= 'Navigation.' <activeLinkName> (<parameterExpression>)

<loadElement> ::= 'LoadElement.' <activeNodeName> (<parameterExpression>)

<parameterExpression> ::= ['NM.' <activeNodeName> '*'? <parameterName>] |
['NM.CurrentPath' <variableName>]

```

#### *Condition Expressions*

```

<condition> ::= (<boolExpression>) (<booloperator> (<boolExpression> )*)

```

```

<boolExpression> ::= <operand> <compOperator>
(<value> | <operand> | <literal>) | 'not' <boolExpression>

<foreachExp> ::= 'Foreach' (<variableName> ',') + 'in' ( (<PE> |
<conceptName> ) , ) + 'do'

<operand> ::= <PE> ((<arithmeticOperator>) (<PE> | <number>)) ?

<PE> ::= ['UM.User' | 'DM.<conceptName>' | <parameterName>] {'.
'<targetRole>'}* ','
[<attributeName> | <conceptName>]

<PE> ::= 'NM.<nodeName>'. '{<linkName>'}* '.' (<attributeName> |
<nodeName>)? | <parameterName>{'.'<attributeName>}

<compOperator> ::= '<' | '>' | '=' | '<=' | '>=' | '<>'

<booloperator> ::= 'and' | 'or' | 'not'

<arithmeticOperator> ::= '+' | '-' | '*' | '/'

```

### Action

```

<action> ::= ';' <action> | <setContent> | <selectInstance> |
<selectAttribute> | <hideLink> | <sort> | <addLink> | <addFilterToLink>
| <deleteLink> | <groupUser> | <attachRuleToGroup>

<setContent> ::= 'setContent('(<PE> | <operand>) ', ' (<PE> | <operand> |
<literal>)? ')'

<selectInstance> ::= 'selectInstance('(<PE> | <parameterName> ', '
<variableName> ')

<selectAttribute> ::= 'selectAttribute('(<PE>')

<hideLink> ::= 'hideLink(NM.<linkName>)'

<sort> ::= 'sort NM.<nodeName> 'orderBy' <PE> ['ASC'|'DESC'] 'LIMIT'
<integer> 'Where' <condition>

<addLink> ::= 'addLink('<linkName>', '<origin> ', '<target>')'

<addFilterToLink> ::= 'addFilterToLink('<filter>', '<linkName>')'

<deleteLink> ::= 'deleteLink(' <origin> ', '<target>')'

<groupUser> ::= 'groupUser(UM.User' ', '<groupName>')'

<attachRuleToGroup> ::= 'attachRuleToGroup(' <ruleName> ', '<groupName>')'

```

### Miscellaneous

```

<ruleName> ::= <string>

```

```

<pfrequency> ::= <string>
<activeLinkName> ::= <string>
<activeNodeName> ::= <string>
<parameterName> ::= <string>
<variableName> ::= <string>
<conceptName> ::= <string>
<targetRole> ::= <string>
<attributeName> ::= <string>
<linkName> ::= <string>
<nodeName> ::= <string>
<groupName> ::= <string>
<filter> ::= <string>
<origin> ::= 'NM .' <string> | 'homepage'
<target> ::= 'NM .' <string> | 'homepage'
<value> ::= <number>+ | <boolean>

```

### Data Types

```

<number> ::= '-'? ( <integer> | <float> | <exponential> )
<integer> ::= <nonzero-digit> <digit> *
<float> ::= <integer> '.' <digit> <digit>*
<exponential> ::= <digit> '.' <digit> <digit>* 'E' ('+' | '-' ) <integer>
<digit> ::= '0' | <nonzero-digit>
<nonzero-digit> ::= '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
<boolean> ::= 'true' | 'false'
<string> ::= <char>+
<literal> ::= '"' ( <string> | ' ' | <number> | ',' | '%' ) * '"'
<char> ::= 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' | 'j' | 'k' |
'l' | 'm' | 'n' | 'o' | 'p' | 'q' | 'r' | 's' | 't' | 'u' | 'v' | 'w' | 'x' | 'y'
| 'z' | 'A' | 'B' | 'C' | 'D' | 'E' | 'F' | 'G' | 'H' | 'I' | 'J' | 'K' | 'L'
| 'M' | 'N' | 'O' | 'P' | 'Q' | 'R' | 'S' | 'T' | 'U' | 'V' | 'W' | 'X' | 'Y' |
'Z'

```



Universitat d'Alacant  
Universidad de Alicante

## Appendix C

### PRML Translator

This appendix describes the PRML translator, a prototype tool to implement transformations from personalization specifications in PRML to the specifics of a personalization specification in a target Web modelling approach. The obtained translation can be used within the CASE tool of the target methodology.

It currently supports the transformations defined in Chapters 7 and 8 from PRML to the specifics of the UWE and Hera approaches.

#### C.1 The PRML Translator Implementation

In order to implement the PRML Translator the ANTLR [ANTLR] parser generator tool has been used. ANTLR is a language tool that provides a framework for constructing recognizers, compilers and translators from grammatical descriptions containing Java, C#, C++ or Python actions. ANTLR allows to generate lexical, syntactic and semantic analyzers in several languages from a set of files written in an own notation. That notation is basically a set of EBNF rules and a set of auxiliary constructors.

ANTLR generates pred-LL(k) analyzers, and uses a pred-LL(k) analyzer to read the files with the EBNF rules. ANTLR allows actions in its rules, besides other features like parameters, values return or grammar inheritance. We use this application to generate a lexical analyzer (lexer) which will return the tokens sequence, and a syntactic analyzer (parser) which will check the right order of the tokens. We do not generate a semantic analyzer, we only check that the rule is well formed. We choose Java as the language in which ANTLR generates the different analyzers. In Figure C-1 a schema of how the PRML translator works and its relation with the ANTLR generator is shown.

The ANTLR takes two files as input: the first one corresponds with the lexical analyzer (extending the lexer class) where all the tokens of PRML are

defined. The lexic analyzer reads the characters one by one and groups them in tokens which will constitute the input for the next stage of the translator. There are two types of tokens: PRML keywords (i.e. When, foreach...) and not specific tokens like labels or constants.

The second file corresponds with the syntactic analyzer and contains all the EBNF grammar of PRML with the translation rules for each of the elements of the grammar to the specifics of the target methodology (this means that there should be a different file with a set of transformation rules for each of the target approaches). The syntactic analyzer (i.e. parser) gets as an input the tokens from the lexical analyzer (the parser does not work directly with characters) and checks if those tokens are arriving in the right order (the order allowed by PRML). The output of the syntactic analysis is a syntactic tree. The functions of the parser are: accept what is syntactically correct and reject what is not, make explicit the hierarchic order of the operators in PRML and guide the process of translation (syntactic driven translation).

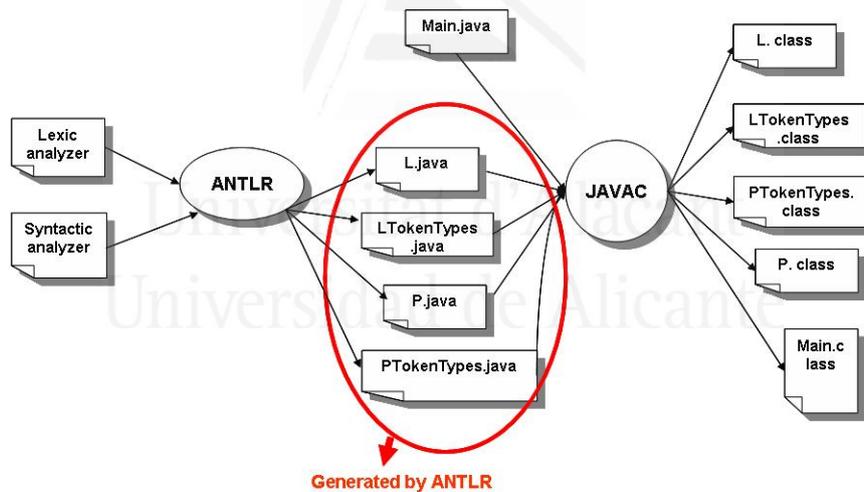


Figure C-1: PRML Translator behaviour

When executing the ANTLR with those two files as an input, the output is a set of files with source code (in our case Java). Once the code is generated it has to be compiled (only once) and it is ready for its execution.

## C.2 Execution Example

The PRML Translator Engine has been implemented as a Web application implemented as a servlet, in which the input is a rule (or set of rules) in PRML, and the output is the translation of that(those) rule(s) in the language specified. The execution architecture can be seen in Figure C-2.

We have already performed (successfully) the transformations from PRML Lite to the specifics of the UWE methodology, which uses OCL [UML 2.0 OCL specification] to express the adaptation rules, and to the specifics of the HPG engine [Fransincar et al, 2006], which is the engine for presentation generation of the Hera [Fransincar and Houbenl, 2002; Houben et al, 2004] Web design method and expresses the adaptation in form of SeRQL [OpenRDF] queries.

The URL of the PRML Translator is <http://gplsi.dlsi.ua.es/traductor/traductor.jsp>. A screenshot of the tool is shown next (Figure C-3).

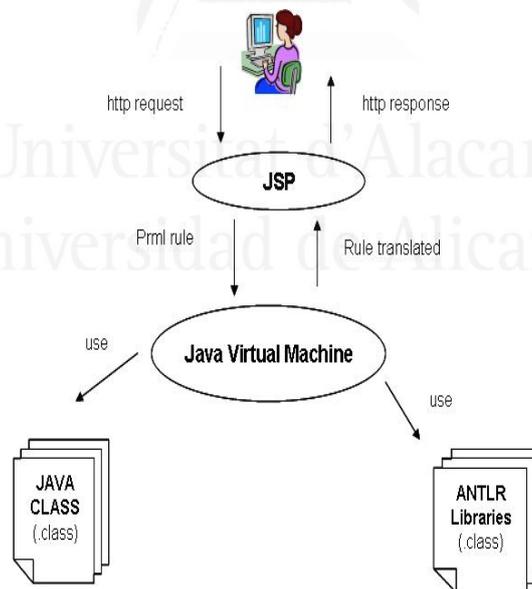


Figure C-2: Execution Architecture

In the upper part the PRML rule or set of rules are introduced. Then the user chooses if the translation is done to Hera or to UWE and the translated rule (or set of rules) is shown.

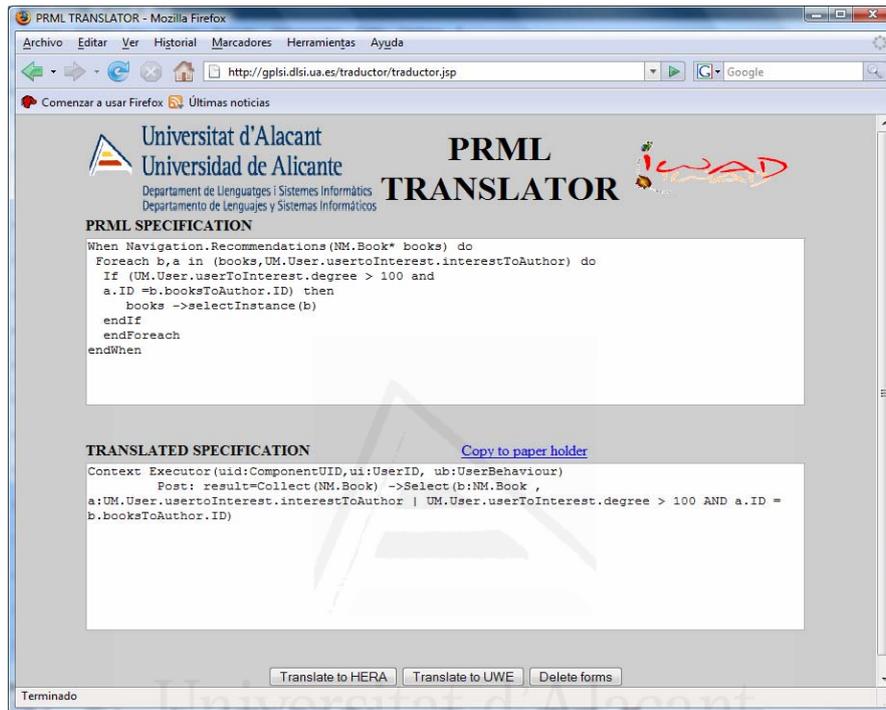


Figure C-3: Screenshot of the PRML Translator

### C.3 Conclusions and Further Work

The PRML Translator Engine is a prototype tool for automatic translation from PRML rules to the specifics used for defining personalization in a target Web methodology. This tool helps to reduce the learning curve of the personalization designer avoiding from learning the specifics of different proprietary solutions and abstracting him from the programming code. In this way a PRML specification can be reused among different Web modelling approaches. The output of the translation can be used within the tool of the target Web modelling method.

Future work includes the translation from PRML to other target approaches like WebML or OOHDM. Moreover, improvements of the tool are currently being done like the concurrent access of several users.



Universitat d'Alacant  
Universidad de Alicante



Universitat d'Alacant  
Universidad de Alicante

# Appendix D

## Spanish summary

En este apéndice se presenta un resumen en español del presente trabajo. Para ello en cada una de las secciones se resumen las ideas principales de cada capítulo de la tesis, excepto del capítulo del estado del arte.

### D.1 Introducción

#### D.1.1 Motivation

La World Wide Web ha cambiado la forma en que nos comunicamos, consultamos y compartimos información. Tiene un gran impacto en nuestra vida diaria, influyendo en la forma que trabajamos, educamos, comerciamos, etc.

En la última década, ha crecido rápidamente el número y complejidad de los sitios Web y la cantidad de información que ofrecen. Nos enfrentamos a un espectro nuevo y amplio de aplicaciones Web que hace que surjan nuevos retos y requisitos que van más allá de la especificación del mapa de navegación, incluyendo aspectos como la evolución continua de los sitios Web junto con la existencia de audiencia heterogénea que implica que el diseño de los sitios Web debería tratar diferentes usuarios, necesidades, preferencias y capacidades. Además los sitios Web normalmente sirven a una gran audiencia, lo cual puede llevar a problemas de mantenimiento y usabilidad [Bevan, 2005; Nielsen, 1992].

Además, la complejidad de la WWW implica problemas de navegación y comprensión. Los sistemas Web tradicionales estaban basados en la aproximación “uno para todos” en los que se presenta la misma información del mismo modo para todos los usuarios. Esta aproximación tiene efectos negativos como la incapacidad de satisfacer las necesidades heterogéneas, preferencias y objetivos de la audiencia. El desarrollo ad-hoc de sistemas basados en Web necesita una aproximación sistemática y un control de

calidad. La Ingeniería Web, una nueva disciplina emergente, aboga por un proceso y una aproximación sistemática para desarrollar sistemas basados en Web de alta calidad. En este contexto surgieron las metodologías de diseño Web [Casteleyn et al, 2003; Ceri et al, 2003; Gómez et al, 2001; Houben et al, 2004; Kappel et al, 2002; Koch, 2001; Rossi et al 2001], proveyendo soluciones a los diseñadores (soporte para el diseño, ayuda en estructuración consistente, mantenimiento más sencillo) y para usuarios (contenido mejor estructurado, navegación más sencilla).

Para acomodar mejor al usuario individual, es necesario desarrollar sistemas Web personalizados (PWS), que son sistemas con la capacidad de adaptar su comportamiento a los objetivos, tareas, capacidades y necesidades de los usuarios individuales (y grupos de usuarios). Hay muchas definiciones dadas para personalización, [Cingil et al, 2000., Blom, 2000., Kim, 2002, Wang and Lin, 2002], sin embargo no hay una definición universal. La meta de la personalización es incrementar la relevancia del sistema al individuo. La personalización Web se define aquí como el proceso de cambiar el contenido y estructura del sitio Web a las necesidades, objetivos, intereses y preferencias de cada usuario haciendo uso del comportamiento de navegación del usuario.

Al diseñar sitios Web personalizados (PWS), el diseño ad-hoc es imposible. Muchos métodos de modelado Web proveen una aproximación conceptual y sistemática para el diseño complejo y la implementación de aplicaciones Web. La mayoría de estas metodologías permiten desarrollar PWS donde las especificaciones de personalización están embebidas en sus modelos de diseño. Sin embargo, cuando la personalización está fuertemente ligada con el resto de la aplicación y la tecnología subyacente, surgen algunos problemas como la dificultad de mantenimiento de la personalización y la incapacidad de reutilizar las especificaciones de personalización entre diferentes aproximaciones.

Es obvio que muchas aproximaciones existentes están basadas en reglas y ofrecen medios potentes para especificar la personalización, así como herramientas (ej. ILog JRules, LikeMinds, WebSphere, Rainbow,...) que hacen que el usar políticas de personalización sea más sencillo. Sin embargo, en la mayoría de los casos, las especificaciones no son portables, porque no sólo usan diferentes técnicas de modelado sino que también diferentes métodos de implementación. Otros problemas de estas aproximaciones son el bajo nivel de abstracción y la insuficiente separación de la especificación

funcional de la aplicación, lo que puede causar problemas de reutilización y dificultar el mantenimiento y escalabilidad de las aplicaciones personalizadas resultantes.

La forma de modelar la adaptación varía en las diferentes metodologías, algunas utilizan condiciones para especificar alternativas de diseño, otras usan reglas para especificar el comportamiento adoptivo, otros métodos usan queries sobre el modelo de datos, etc. Estas diferencias hacen que el diseñador de un PWS elija una u otra de esas aproximaciones dependiendo de sus necesidades particulares y preferencias. Esto puede causar diversos problemas al diseñador:

*P1. No reutilización*

Una estrategia de personalización especificada con un método de diseño Web A no puede ser reutilizada al diseñar un sitio Web con un método de diseño diferente B. Esto se debe a varios factores:

- La especificación de la personalización no es independiente de la tecnología subyacente.
- Los diferentes métodos de diseño soportan acciones de personalización distintas.

*P2. Semántica restringida: Limitaciones*

Todas las aproximaciones existentes tienen limitaciones para especificar la personalización. Por ejemplo algunas de ellas no consideran la adaptación de la presentación, otras no consideran añadir o borrar enlaces dinámicamente. El diseñador puede tener problemas al decidir que aproximación utilizar para dar soporte a los requisitos de personalización del sitio Web.

*P3. No herramienta CAWE*

La mayoría de las aproximaciones no proveen de una herramienta CAWE<sup>1</sup> subyacente en Ingeniería Web y todavía menos para soportar el modelado de la personalización. La falta de herramientas de este tipo hace

---

<sup>1</sup> Computed Aided Web Engineering

que la personalización se implemente en una manera ad-hoc. Además, al no disponer de una herramienta las diferentes metodologías adaptivas no están testeadas correctamente.

La consecuencia es que el diseñador debería elegir las aproximaciones con una herramienta de generación.

#### *P4. Complejidad*

La personalización está considerada una disciplina completa, no sólo relacionada con la Web, y está presente en muchos campos diferentes como la televisión [Chorianopoulos and Spinellis, 2002], educación [Apple et al, 2004], medicina [The Personalized Medicine Coalition], etc. En el caso de la Ingeniería Web las personas responsables de la personalización pueden tener conocimientos de psicología y sociología y no deben ser necesariamente programadores con experiencia. Por ello creemos que de forma análoga a la figura del *diseñador de presentación*, es necesario definir un rol de un diseñador responsable de definir la personalización en el sitio Web tratándolo como un aspecto ortogonal, *un diseñador de personalización*.

Las diferentes metodologías Web utilizan diversos lenguajes para especificar la personalización. Al definir un PWS, dependiendo del método de diseño utilizado, el *diseñador de personalización* debe aprender un lenguaje diferente para especificar la personalización. Esto puede ser una tarea pesada para el diseñador, que como hemos mencionado antes, no tiene porque ser un programador con experiencia. Además:

- Algunos de estos lenguajes son difíciles de aprender y de usar.
- Estos lenguajes tienen una sintaxis demasiado específica ligada a los elementos de modelado de la aproximación, lo que hace oscuro su entendimiento y comparación con otros lenguajes de personalización.
- La mayoría de estos lenguajes no son específicos de dominio (de la personalización).

#### *P5. Personalización: solo definida en tiempo de diseño*

Otro problema para el diseñador puede ser el tener que añadir personalización a un sitio Web que ya está generado. Utilizar una aproximación de modelado Web donde la personalización está embebida en la metodología, implica modificar todos los modelos para añadir adaptividad al sitio Web y regenerar las páginas Web.

Es difícil predecir en tiempo de diseño quienes serán los usuarios del sitio Web y cómo usaran el sitio Web para definir una estructura óptima de la aplicación Web. Para ello es necesario disponer de información en tiempo de ejecución sobre el uso del sitio Web. Esta información puede usarse para personalizar para un usuario particular o grupos de usuarios.

Los diseñadores Web pueden beneficiarse de esta información actuando según el comportamiento de usuario, especificando en tiempo de diseño que adaptación será posible en tiempo de ejecución. El objetivo de esta tesis es permitir la especificación en tiempo de diseño de comportamiento personalizado en tiempo de ejecución para sitios Web.

### **D.1.2 Propuesta**

Para solucionar los problemas descritos, proponemos tratar la personalización como un aspecto ortogonal. La personalización se considera un aspecto separado del sitio Web, independiente de la tecnología subyacente, permitiendo el reuso de especificaciones de personalización. La solución propuesta está basada en un lenguaje de reglas de alto nivel llamado PRML (*Personalization Rules Modeling Lenguaje*) [Garrigós et al, 2005; Garrigós and Gómez, 2006] que el diseñador puede usar para especificar en tiempo de diseño la personalización a ejecutar en tiempo de ejecución. Se definen dos niveles de conformidad en este lenguaje:

- *PRML Lite*: Este nivel comprende las operaciones básicas soportadas por los métodos de modelado Web más representativos [Casteleyn et al, 2003; Ceri et al, 2003; Gómez et al, 2001; Houben et al, 2004; Koch, 2001; Rossi et al, 2001]. Permite la independencia de la metodología de modelado Web, y por tanto el reuso. El propósito de la definición de este lenguaje es ser capaz de especificar una política de personalización universal y reutilizable.

- *PRML Full*: El propósito de este nivel es dar soporte para acciones de personalización específicas, limitando por tanto la reusabilidad. Con los constructores definidos en este nivel, se pueden especificar estrategias de personalización más complejas. PRML Lite es un subconjunto de PRML Full; por tanto las particularidades definidas en PRML Lite son también válidas en PRML Full.

Este lenguaje trata los problemas antes descritos del siguiente modo:

### *51. Reusabilidad*

La reusabilidad en PRML es posible de dos formas diferentes: por un lado, el lenguaje puede ser reutilizado en diferentes aproximaciones y por otro lado, una estrategia de personalización puede ser reutilizada entre diferentes metodologías Web.

- La reutilización del lenguaje entre diferentes metodologías Web es posible. Para lograr este objetivo PRML ha sido definido como un lenguaje independiente del método. PRML nació en el contexto del método de diseño OO-H [Gómez et al, 2001] para extenderlo con soporte de personalización. Después ha evolucionado para ser un lenguaje genérico independiente de la tecnología subyacente. Para conseguir esta independencia es necesario identificar un denominador común de las metodologías de diseño Web más representativas. Esto se ha realizado abstrayendo los elementos básicos de modelado sobre las que actúan las primitivas y técnicas de personalización. PRML se ha desarrollado usando estos elementos de modelado abstractos que se explican en el Capítulo 4.
- La reutilización de la personalización entre diferentes aproximaciones también es posible si los métodos de diseño Web definen sus modelos de acuerdo con los requisitos especificados en el Capítulo 4. Estos requisitos definen los modelos abstractos, que son el Modelo de Dominio, el Modelo de Usuario y el Modelo de Navegación sobre los que PRML puede actuar. En el Capítulo 4 también se explica el modo de acceder a los elementos de los modelos abstractos de PRML. Para poder usar PRML y reutilizar la misma estrategia de personalización, estas aproximaciones deben tener modelos equivalentes para especificar una aplicación Web. Hemos identificado para cada

aproximación diferente un subconjunto de sus modelos. Podemos garantizar que la expresividad de este subconjunto de modelos corresponde con la de los modelos abstractos definidos en PRML. Por lo tanto, estos subconjuntos de modelos de las diferentes aproximaciones tienen expresividad equivalente. En esta tesis, las aproximaciones de UWE y Hera se usan para mostrar este hecho. Los capítulos 7 y 8 explican como los modelos abstractos se mapean a Hera y UWE, teniendo la misma expresividad. Estos capítulos presentarán transformaciones formales de PRML a UWE y Hera usando el lenguaje estándar QVT [OMG QVT] para mostrar la portabilidad de la aproximación PRML y su reusabilidad entre diferentes metodologías. Para ilustrar este hecho, se presentan ejemplos concretos de transformaciones de reglas PRML a Hera y UWE, considerando los modelos definidos en el caso de estudio que se presenta en el Capítulo 4. Finalmente, en el apéndice C se presenta una herramienta que automáticamente ejecuta estas transformaciones.

### S2. Personalización Avanzada

Como se ha descrito previamente, con PRML Full se puede especificar personalización más compleja (limitando la reusabilidad).

Por ejemplo, además de soportar la personalización en base de comportamiento simple de usuario (como el click del usuario en un enlace), creemos que es importante ser capaz de detectar acciones del usuario más complejas (por ej, una secuencia de activaciones de enlaces). La consecuencia es que la complejidad de las estrategias de personalización a definir (basadas en el comportamiento de navegación del usuario) aumenta.

### S3. Herramienta CAWE

En nuestra propuesta damos soporte a PRML con AWAC, una herramienta prototipo CAWE para la generación automática de aplicaciones Web adaptivas basadas en la metodología A-OOH. A-OOH (Adaptive OO-H) es una extensión de la aproximación OO-H para soportar el modelado de sitios Web personalizados. A-OOH permite modelar el contenido, estructura, presentación y personalización de una aplicación Web. La herramienta AWAC usa los modelos de diseño de A-OOH de la aplicación Web adaptiva a generar como *input*. Una vez

generada, la aplicación Web adaptive contiene además tres módulos para manejar la personalización, los cuales analizan en tiempo de ejecución los eventos de navegación y adaptan el sitio Web según las reglas de personalización lanzadas. Estas reglas de personalización se especifican en un fichero independiente y pueden ser actualizadas sin tener que modificar el resto de la lógica de la aplicación.

#### S4. Usabilidad

PRML es un lenguaje de alto nivel, simple y eficiente. Es un lenguaje específico de dominio [Mernik et al, 2005] así que las reglas definidas con él son muy intuitivas y fáciles de aprender. La curva de aprendizaje del *diseñador de personalización* se reduce debido a dos razones. La primera es que el diseñador se abstrae del código de programación y la segunda es que el diseñador puede definir la personalización usando PRML y después transformarlo al método de diseño a utilizar. Utilizando PRML Lite aseguramos que la especificación de personalización puede mapearse a cualquiera de las aproximaciones existentes, aunque se pueden definir mappings con PRML Full, limitando la reusabilidad entre metodologías.

#### S5. Personalización: definida en tiempo de diseño y en tiempo de ejecución

El objetivo principal de PRML es proveer al diseñador con una manera de especificar estrategias de personalización en tiempo de diseño (que se ejecutarán en tiempo de ejecución).

La personalización se ha convertido en un aspecto clave en el desarrollo Web. Los diseñadores muchas veces son forzados a añadir personalización a un sitio Web existente (en tiempo de ejecución). Debido a la separación de conceptos, PRML también permite al diseñador añadir personalización a un sitio Web ya generado. La especificación de la personalización sería un modulo independiente que puede ser integrado con el resto de modelos sin necesidad de modificarlos.

La independencia de PRML del resto de la lógica de aplicación se consigue utilizando reglas ECA [Ying et al, 2007; Heimrich and Specht, 2003; Chakravarthy, Le, and Dasari, 1999; Dayal, 1988]. El disparo de eventos permite una mejor separación de las especificaciones de personalización de la especificación de la lógica de la aplicación.

Además, utilizar reglas ECA es una decisión natural conforme a los requisitos iniciales del modelo de personalización, donde las acciones reaccionan ante los eventos concretos causados por los usuarios.

### D.1.3 Contribuciones

Las principales contribuciones de este trabajo son las siguientes:

- OO-H se ha extendido para soportar personalización convirtiéndose en A-OOH (Adaptive OO-H, Capítulo 3). Las principales diferencias con respecto a OO-H son:
  - Los sistemas adaptivos hypermedia son sistemas complejos que requieren un proceso de ingeniería de software apropiado para su desarrollo. Esta es la razón por la que el proceso de diseño de A-OOH está basado en el Proceso Unificado (UP) y no en el modelo espiral como el proceso de diseño de OO-H.
  - El Modelo de Navegación ha sido modificado separando las características de presentación que estaban mezcladas en el Modelo de Navegación de OO-H. Además se ha definido un perfil UML para poder representar el Modelo de Navegación utilizando notación UML.
  - Se ha añadido un Modelo de Presentación. Este modelo también utiliza notación UML.
  - Se han añadido un Modelo de Usuario y un Modelo de Personalización para poder definir aplicaciones Web adaptivas.
  - Se han definido varios perfiles UML para los diferentes modelos de A-OOH.
- Se ha definido un lenguaje de alto nivel basado en reglas ECA para especificar personalización. Este lenguaje puede usarlo el diseñador para especificar en tiempo de diseño la personalización a ejecutar en tiempo de ejecución (Capítulos 4 y 5).
- Detección de comportamiento complejo de usuario para la especificación de estrategias de personalización más avanzadas (definido en el Capítulo 5).

- Se ha implementado un prototipo de herramienta para la generación automática y manejo de aplicaciones Web adaptivas basadas en la metodología A-OOH (Capítulo 6).
- Reusabilidad de estrategias de personalización utilizando PRML Lite. Transformaciones de PRML a diferentes aproximaciones de modelado Web (Capítulos 7 y 8).
- Se ha implementado una herramienta para implementar esas transformaciones automáticamente (Apéndice C)

#### **D.1.4 Estructura de la tesis**

La tesis está estructurada del siguiente modo:

El *Capítulo 2* presenta los fundamentos de la investigación descrita en este trabajo. Presenta los conceptos de hipertexto e hipermedia, mostrando los diferentes modelos utilizados para hipertexto, especialmente el conocido modelo Dexter. Además se centra en los sistemas hipermedia adaptivos introduciendo los conceptos principales relativos a la hipermedia adaptiva, métodos y técnicas de adaptación. También describe diferentes métodos de diseño Web que soportan adaptación.

El *Capítulo 3* describe el método A-OOH. En primer lugar se presenta el proceso de diseño propuesto por A-OOH. A continuación se explican los diferentes modelos de la aproximación A-OOH. Finalmente se presentan las conclusiones del capítulo junto con las principales extensiones hechas al método OO-H.

El *Capítulo 4* explica los fundamentos de PRML Lite. El capítulo comienza presentando el caso de estudio usado a lo largo del presente trabajo. PRML Lite se explica en términos de los eventos soportados, la forma de definir condiciones sobre los datos del Modelo de Usuario, y las acciones de adaptación soportadas. También se explica como especificar el orden de ejecución de las diferentes reglas PRML.

El *Capítulo 5* presenta PRML Full extendiendo el caso de estudio usado en el capítulo anterior con requisitos de personalización más complejos. Se

explican los fundamentos de PRML Full presentando su metamodelo y explicando el soporte de navegación compleja del usuario y las diferentes acciones de adaptación soportadas por el lenguaje. Además se provee al diseñador de una guía de diseño para poder definir reglas con PRML Full adecuadamente.

El *Capítulo 6* presenta AWAC, un prototipo de herramienta CAWE para la generación automática de aplicaciones Web adaptivas basadas en la metodología A-OOH. Esta herramienta toma como entrada los modelos de diseño A-OOH del sitio Web adoptivo a generar. Una vez generado, el sitio Web adoptivo también contiene tres módulos para manejar la personalización en tiempo de ejecución, analizando los eventos de navegación del usuario y adaptando el sitio Web según las reglas de personalización lanzadas.

Los *Capítulos 7 y 8* describen como exportar PRML a los métodos de diseño UWE y Hera. La implementación de dichas transformaciones se presenta en el apéndice C de la presente tesis.

El *Capítulo 9* presenta las conclusiones de la presente tesis. Se provee un resumen explicando las contribuciones y limitaciones del trabajo. Finalmente se discuten posibles extensiones y trabajo futuro.

## **D.2 Requisitos para utilizar PRML Lite**

El propósito principal de PRML Lite es poder especificar la personalización de una manera independiente del método y portable o reutilizable por otras aproximaciones. Para lograr este objetivo, las reglas PRML se definen como un componente separado que puede ser conectado o mapeado a diferentes aproximaciones de modelado Web. Para definir PRML Lite se siguieron los siguientes pasos:

- *Identificar un denominador común de un conjunto de lenguajes de modelado existentes y abstraer el conjunto de conceptos fundamentales.*

Para definir adecuadamente un lenguaje independiente del método, el primer paso es identificar un denominador común de un conjunto de lenguajes de modelado existentes y abstraer el conjunto de conceptos fundamentales. Se han estudiado diferentes métodos de modelado con soporte de la personalización. Las aproximaciones consideradas en este estudio han sido WebML [Ceri et al, 2003], OOHDM [Rossi et al, 2001], UWE [Koch, 2001], Hera [Houben et al, 2004] y WSDM [Casteleyn et al, 2003]. Este conjunto de aproximaciones es suficientemente representativo para mostrar la genericidad de PRML Lite. Todas ellas tienen soporte de personalización excepto WSDM, el cual soporta adaptación para todos los usuarios pero podría ser fácilmente extendido para soportar personalización. Se definieron algunos requisitos esenciales para poder utilizar las especificaciones de personalización de PRML Lite en un método de modelado Web:

**R1:** Un Modelo de Dominio (DM) debe formar parte de la especificación de una aplicación Web. Debe haber una forma de especificar la estructura de datos del sitio Web. El DM ha de cumplir el siguiente requisito:

**R1.1:** El DM debe estar compuesto de conceptos, atributos y relaciones entre conceptos.

**R2:** Un Modelo de Navegación (NM) debe ser parte de la especificación de la aplicación Web. Debe existir un mecanismo para acceder a los datos de contenido de los objetos de navegación. Este modelo ha de cumplir el siguiente requisito:

**R2.1:** El NM debe tener nodos y enlaces como componentes principales. Los nodos son vistas restringidas de los conceptos del dominio pero también pueden ser nodos estáticos sin conceptos asociados. Cada nodo puede tener asociado un concepto del DM o el UM. Los nodos contienen los atributos y operaciones que van a ser mostrados en la navegación.

Los requisitos R1 y R2 aseguran que el objeto de las reglas de personalización exista y el contenido referenciado por la condición y acción de las reglas. La especificación del dominio y la navegación están típicamente presentes en las aproximaciones de modelado Web.

**R3:** Un Modelo de Usuario (UM) debe ser parte de la especificación de la aplicación. Una alternativa puede ser la especificación de un espacio de datos actualizable. Debe existir un mecanismo para acceder el contenido de los objetos del Modelo de Usuario.

Este requisito asegura que el objeto de las reglas de adquisición (actualización de datos) exista y que pueda ser adecuadamente referenciado. Todos los métodos modernos soportan algún tipo de personalización o al menos permiten asociar las acciones del usuario con actualizaciones de los datos.

**R3.1:** Análogamente al DM, el UM debe estar compuesto de conceptos, atributos y relaciones entre conceptos.

**R4:** Se deben poder utilizar eventos y condiciones con el NM y el UM para lanzar y ejecutar la política de personalización deseada.

El requisito R4 asegura que la personalización se puede lanzar mediante eventos soportados en las diferentes metodologías. Las acciones de personalización se ejecutarán dependiendo de la evaluación de la condición.

**R5:** Además de los requisitos previos, debe ser posible definir acciones sobre los elementos de los modelos.

El requisito R5 asegura que las acciones de PRML tienen su correspondencia en acciones que pueden ser especificadas usando diferentes métodos.

- *Desarrollar un lenguaje de reglas común sobre los conceptos abstractos*

Una vez se han abstraído los modelos y los conceptos sobre los que PRML debe operar, se deben definir algunos aspectos para desarrollar un lenguaje de reglas común:

- *Momento de la adaptación: ¿cuándo personalizar?* Se deben definir los eventos soportados en PRML para lanzar las reglas.
- *Condiciones.* Se debe especificar la forma de hacer referencia a los conceptos de los diferentes modelos y los operadores de comparación soportados en PRML.
- *Operaciones sobre conceptos.* Finalmente, se deben describir las operaciones permitidas en PML sobre los diferentes conceptos.

PRML Lite está basado en un metamodelo MOF [OMG] (ver Figura D-1) que define el conjunto de constructores del lenguaje y las diferentes partes que forman una regla PRML Lite así como los distintos eventos y acciones soportadas. El elemento principal del metamodelo es la metaclasses *rule* que representa el concepto de regla conteniendo los elementos que la definen. Los

elementos que definen una regla son los que representan su estructura principal y son explicados a lo largo del Capítulo 4.

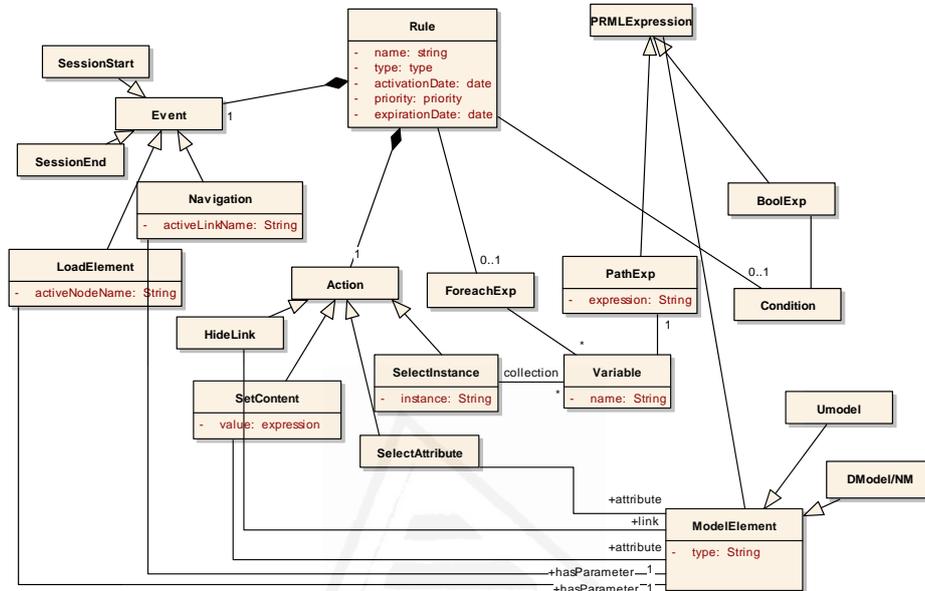


Figura D-1: Metamodelo de PRML Lite (simplificado)

La estructura básica de una regla definida con PRML es la siguiente:

```

When event do
    [Foreach expression]
        If condition then action [else action] endif
    [endForeach]
endWhen
    
```

Una regla PRML está formada por un evento y el cuerpo de la regla, que contiene una condición (opcional) y una acción para ser ejecutada. La parte del *evento* define cuando debe ser disparada la regla. Una vez disparada la *condición* se evalúa (si existe) y en el caso de ser evaluada positivamente, la *acción* es ejecutada. La condición también puede tener una cláusula *else*. También puede existir una *expresión Foreach* en la regla cuando la acción y la condición actúan sobre un conjunto de instancias.

Para satisfacer un requisito de personalización a menudo debemos definir cómo actualizar el conocimiento sobre el usuario (mediante una regla de

adquisición). Esta información se almacena en el espacio de datos del Modelo de Usuario. Además, debemos definir los efectos que esta personalización causa al contenido presentado y a la estructura de la navegación (mediante reglas de personalización). Estas reglas utilizan la información especificada en el Modelo de Usuario para describir acciones de adaptación. Dependiendo del objeto de adaptación se consideran dos tipos de reglas de personalización: reglas de personalización de la navegación (para modificar la navegación), y reglas de personalización del contenido (para añadir/borrar/adaptar el contenido).

### D.3 PRML Full

El lenguaje PRML (Personalization Rules Modeling Language) define dos niveles de conformidad: PRML Lite (presentado en el capítulo 4 y en la sección anterior) que se ha definido abstrayendo los conceptos básicos (de adaptividad) de las metodologías Web más representativas, y PRML Full, cuyo propósito es soportar acciones de personalización más complejas y específicas limitando la reusabilidad conseguida con PRML Lite. PRML Lite es un subconjunto de PRML Full. Con PRML Full las metodologías que soporten la especificación de políticas de personalización más ricas que las soportadas por PRML Lite, se pueden beneficiar del uso de PRML. Los beneficios de usar PRML Full son:

- *Fácil de aprender, de leer y mantener por los diseñadores Web:* Del mismo modo que en PRML Lite la curva de aprendizaje del diseñador se minimiza.
- *Posibilidad de seguir el comportamiento complejo de usuario:* PRML Lite ya soporta la personalización basada en comportamiento simple de usuario (como un clic del usuario en un enlace). Sin embargo, es importante ser capaz de soportar la detección de acciones del usuario más complejas. Al hacer esto, algunos problemas deben ser resueltos, como el seguimiento del comportamiento del usuario.
- *Posibilidad de adaptar el sitio Web para un grupo dinámico de usuarios.*

Para poder utilizar PRML Full, las metodologías Web han de soportar los mismos requisitos mínimos especificados para PRML Lite (sección anterior de este apéndice o sección 4.2 del Capítulo 4). PRML Full también está basado en un metamodelo MOF [OMG] (ver Figura D-2) que completa el metamodelo definido para PRML Lite con el conjunto completo de constructores del lenguaje.

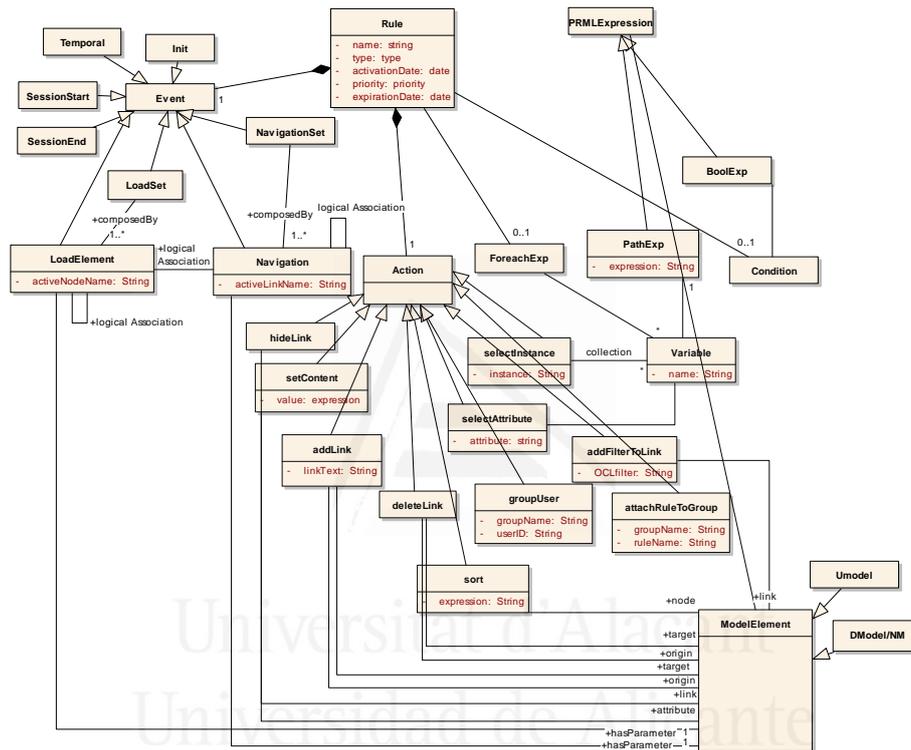


Figure D-2: PRML Full Metamodel

La estructura básica de una regla definida con PRML Full es la siguiente:

```

When event do
    [Foreach expression]
        If condition then
            action1;
            action2; ...
            actionn
        [else
            action1;
            action2; ...
            actionn ]
        endif
    [endforeach]
endWhen
    
```

La diferencia con una regla definida con PRML Lite es que en la misma regla podemos definir varias acciones a ejecutar. Esto puede reducir el número de reglas necesarias para definir una estrategia de personalización y permite ejecutar acciones sucesivas sobre el mismo elemento del modelo. Sin embargo, el diseñador de personalización ha de seguir algunas guías de diseño para escribir reglas correctas. Estas reglas de diseño se explican en la Sección 5.2.3 del Capítulo 5.

#### D.4 AWAC

El propósito principal de la herramienta AWAC es la generación automática de una aplicación Web adaptiva desde los modelos de A-OOH. La herramienta AWAC toma como *input* los modelos de diseño de A-OOH: el *Modelo de Dominio*<sup>2</sup>(DM), en el que se define la estructura de los datos del dominio, el *Modelo de Navegación* (NM) que define la estructura y comportamiento de la vista de navegación sobre los datos del dominio, y finalmente el *Modelo de Presentación* (PM) que define la representación de la aplicación generada. Para ser capaz de modelar la adaptación/personalización en tiempo de diseño se definen el *Modelo de Usuario* (UM), en el que se describe la estructura de la información necesaria para personalización. Típicamente, la información contiene las creencias y conocimiento del sistema sobre el usuario y es la base para las acciones de personalización descritas en el Modelo de Personalización. El *Modelo de Personalización* (PeM) especifica las políticas de personalización. Además de definir la personalización del contenido, la estructura de navegación y la presentación, el Modelo de Personalización también define actualizaciones de los datos contenidos en el *Modelo de Usuario*.

Estos modelos se representan por medio de elementos XML (en formato XMI [XML Metadata Interchange]). El motivo para elegir una representación en XMI de los modelos es que este formato puede ser generado fácilmente desde los modelos UML<sup>3</sup>. Para leer y procesar los modelos de A-OOH para la

---

<sup>2</sup> A veces llamado Modelo Conceptual

<sup>3</sup> La mayoría de las herramientas UML permiten esta transformación.

generación de las páginas Web finales se ha utilizado la tecnología .NET. Esta tecnología nos provee con la clase DOM (XML Document Object Model), con la que podemos representar en memoria los documentos XML.

El *output* de la herramienta AWAC consiste de:

- **La aplicación Web adaptiva generada:** la versión actual de AWAC genera páginas Web ASP.net.
- **Módulos para el manejo de la personalización**
- **Base de datos de la aplicación:** Los modelos de A-OOH representados inicialmente en XMI se mapean a una base de datos orientada a objetos. Dependiendo de las acciones de personalización ejecutadas cada usuario tiene un conjunto diferente de instancias de los modelos de A-OOH. Esta base de datos también contiene información del usuario relative al dominio. La idea de usar una base de datos relacional fue rechazada debido a la complejidad de mapear de orientado a objetos a persistencia relacional. De esta manera la base de datos puede ser generada automáticamente del conjunto de modelos de A-OOH. La tecnología utilizada es db4o [Db4o], la base de datos para objetos de código abierto más popular, native a Java y .NET.

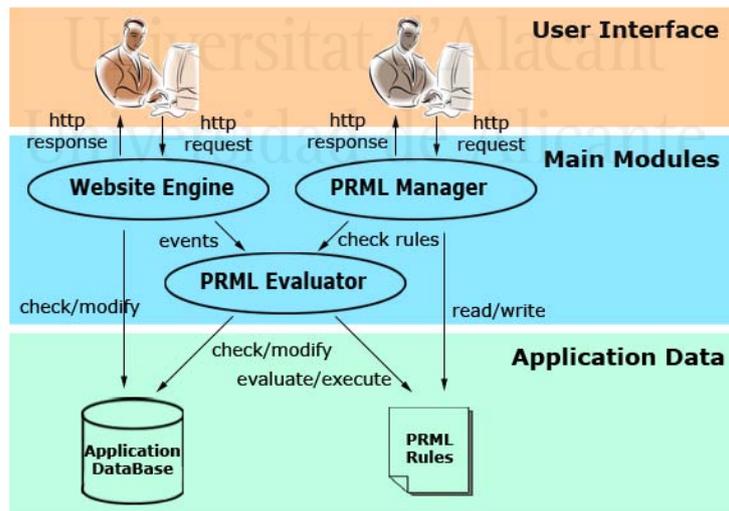


Fig. D-3. Arquitectura de la aplicación generada por AWAC

Como trabajo futuro, además de implementar toda la funcionalidad de PRML, está el añadir una interfaz gráfica de usuario para definir los modelos de A-OOH utilizando AWAC. Ahora, para este propósito y generar los ficheros XMI se utiliza la herramienta Enterprise Architect Design tool [Enterprise Architect] en la que hemos definido los perfiles UML necesarios para modelar los diagramas de A-OOH.

## D.5 Portabilidad a UWE y Hera

En esta sección, se describe brevemente las transformaciones entre PRML y las metodologías UWE y Hera. Las transformaciones han sido definidas usando el lenguaje de transformaciones QVT [OMG QVT] (MOF 2.0 Query/View/Transformation language) que es una aproximación estándar propuesta por OMG. Además es una parte esencial del estándar MDA (Model Driven Architecture) [OMG MDA] como un medio para definir transformaciones formales y automáticas entre modelos.

En la figura D-4, se muestra la secuencia de transformaciones necesarias para diferentes tipos de reglas PRML. Este conjunto de relaciones se aplica conforme a la estructura de una regla PRML (ver Sección 4.3 de la tesis): primero, se aplica la relación *PRMLRule* la cual dirige toda la transformación. De esta forma, la regla decide cual será la próxima relación a aplicar de acuerdo con la expresión en la cláusula “Where”. Por ejemplo, si el evento es *Navigation*, entonces se aplica la relación *PRMLNavigation*. Esta relación (*PRMLNavigation*) llama a la relación *PRMLAction*, ya que la parte más importante de una regla PRML es la acción a ejecutar. La relación *PRMLAction* ejecutará la relación apropiada dependiendo del tipo de acción. Por ejemplo, si la acción es *SelectInstance*, se aplicará la relación *PRMLSelectInstance*. Esta relación junto con las relaciones *PRMLCondition* y *PRMLForEach* se utilizan para crear el modelo de personalización correspondiente de UWE o Hera partiendo de modelo PRML.

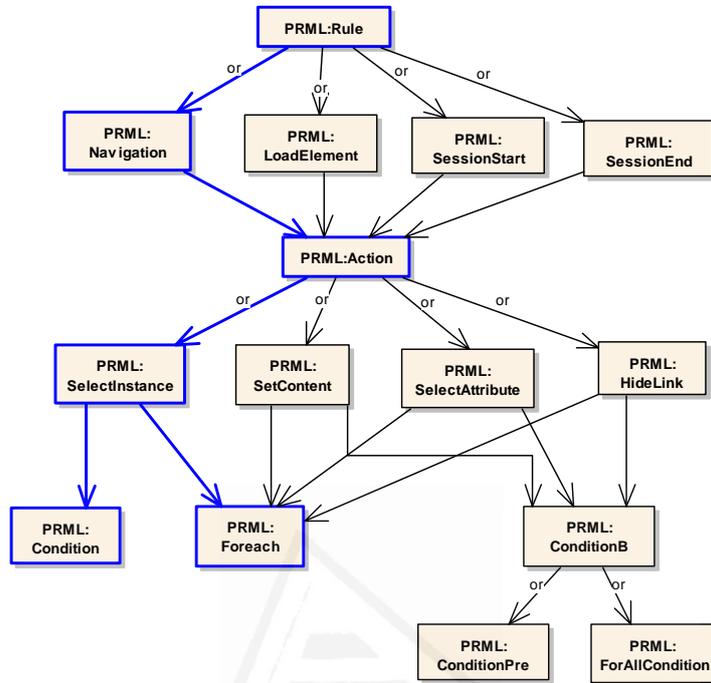


Figure D-4 Secuencia de transformaciones

Como ejemplo, vamos a ver la transformación necesaria para la siguiente regla PRML. Esta regla se define en la Sección 5.3.3 en respuesta al segundo requisito de personalización definido en el caso de estudio (i.e. ofrecer recomendaciones). Esta regla selecciona las instancias a mostrar del concepto libro, los cuales están escritos por un autor en el que el usuario tiene un grado de interés mayor que 100.

PRML	<pre> When Navigation.Recommendations(NM.Book* books) do   Foreach b,a in     (books,UM.User.userToInterest.interestToAuthor) do       If (UM.User.userToInterest.degree &gt; 100 and         a.ID =b.booksToAuthor.ID) then         books-&gt;selectInstance(b)       endIf     endForeach   endWhen </pre>
	En UWE la regla correspondiente es la siguiente:

UWE	<p><b>Context:</b>  <code>Rule::Executor(uid:ComponentUID,ui:UserID,ub:UserBehaviour)</code></p> <p><b>Post:</b> <code>result=Collect(NM.Book)-&gt;Select(b:NM.Book, a:UM.User.usertoInterest.interestToAuthor   b.bookToAuthor.ID=a.ID and UM.User.userToInterest.degree&gt;100)</code></p> <p>En la postcondición de la regla, una expression Select de OCL selecciona las instancias de los libros en los que el usuario tiene un grado de interés mayor que 100.</p> <p>En esta regla podemos ver como se aplican las relaciones QVT descritas en esta sección:</p> <ul style="list-style-type: none"> <li>• <b>PRMLRule:</b> Una regla PRML se transforma en una regla UWE.</li> <li>• <b>PRMLNavigation:</b> Un evento de navegación PRML se transforma en un evento de navegación en UWE. Esta transformación no puede verse en la representación textual porque afectad directamente al metamodelo de adaptación de UWE.</li> <li>• <b>PRMLAction:</b> Una acción PRML se transforma en una expression OCL que representa la regla UWE.</li> <li>• <b>PRML SelectInstance:</b> Una vez se ha comprobado que los elementos que representan la acción <i>SelectInstance</i> existen en la regla PRML, la relación fuerza a que la expression OCL correspondiente contenga los siguientes elementos: una clase <i>IteratorExp</i>, cuyo tipo es “Select”, una colección de elementos, que representan la fuente del <i>IteratorExp</i>, del mismo tipo que la Variable de la acción <i>SelectInstance</i> de PRML.</li> </ul> <p><code>Post: result=Collect(NM.Book)-&gt;Select()</code></p> <ul style="list-style-type: none"> <li>• <b>PRMLForEach:</b> El elemento <i>ForEach</i> (con una variable que tiene un tipo y un elemento <i>PathExp</i>) de la regla PRML se transforma en una expresión <i>IteratorExp</i> que contiene una variable (con su tipo). La variable es el iterador de la expresión <i>IteratorExp</i>.</li> </ul> <p><code>Post: result=Collect (NM.Book)-&gt;Select(b:NM.Book, a:UM.User.usertoInterest.interestToAuthor   )</code></p> <p>.</p>
-----	---

	<ul style="list-style-type: none"> <li>• <b>PRMLCondition:</b> El cuerpo del elemento <i>IteratorExp</i> se define como una expresión OCL que corresponde con la expresión booleana del elemento condición en la regla PRML.   <b>Post:</b> <code>result=Collect(NM.Book)-&gt;Select(b:NM.Book, a:UM.User.userToInterest.interestToAuthor   b.bookToAuthor.ID=a.ID and UM.User.userToInterest.degree&gt;100)</code></li> </ul>
Hera	<p>En Hera el correspondiente conjunto de queries se obtiene aplicando las relaciones QVT del siguiente modo:</p> <ul style="list-style-type: none"> <li>• <b>PRMLRule:</b> Una regla PRML se transforma en un conjunto de queries.</li> <li>• <b>PRMLNavigation:</b> Un evento de PRML <i>navigation</i> se transforma en una query de tipo condición sobre la porción <i>recommendations</i> (<i>SliceCondition_ID1</i>) y en una query de sesión y una de resultado (<i>Query_ID1</i> y <i>QueryResult_ID1</i>) que almacenan el conjunto de libros recibidos como parámetro del evento en la regla PRML como una variable de sesión.</li> <li>• <b>PRMLAction:</b> Una acción PRML se transforma en una query de tipo condición que debe ser construida dependiendo del tipo de acción.</li> <li>• <b>PRML SelectInstance:</b> Una vez comprobado que los elementos que representan la acción <i>SelectInstance</i> existen en la regla PRML, la relación fuerza a que la correspondiente query condición tenga una cláusula <i>SelectClause</i> en la condición de visibilidad actuando sobre una colección de elementos del mismo tipo que la variable de la acción PRML <i>SelectInstance</i>.   <b>SELECT B</b></li> <li>• <b>PRMLForEach:</b> El elemento <i>ForEach</i> (con una variable que tiene tipo y un elemento <i>PathExp</i>) de la regla PRML se transforma en una cláusula <i>from</i> en la query condición con las mismas <i>PathExpressions</i> que en la expresión <i>foreach</i> de PRML.   <b>FROM</b>  <code>{B}&lt;cm:booksToAuthor&gt;(BTA)&lt;um:ID&gt;{ID},  &lt;um:userToInterest&gt;{UI}&lt;um:degree&gt;{D},  &lt;um:userToInterest&gt;{UTI}&lt;um:interestToAuthor&gt;{ITA}&lt;</code></li> </ul>

<pre>um:ID&gt;{ID2} ,   {session}&lt;session:user&gt;{U}</pre> <ul style="list-style-type: none"> <li> <b>PRMLCondition:</b> El cuerpo del elemento <i>IteratorExp</i> se define en la cláusula <i>where</i> de la query <i>select</i> de la condición de visibilidad como una expresión que coincide con la expresión booleana del elemento <i>condition</i> en la regla PRML. </li> </ul> <pre>WHERE D&gt;100 and ID= ID2 and B in Select * from session:books</pre>
<p><b>Queries Session y Result</b></p> <p>Este conjunto de queries rellenan la variable de sesión “book” con el conjunto de libros del modelo de dominio.</p> <pre>&lt;rdfs:Class rdf:ID="Query_ID1"&gt; &lt;rdfs:subClassOf rdf:resource="#&amp;Query" /&gt;   &lt;slice:queryString&gt;     SELECT * FROM &lt;cm:books&gt;   &lt;/slice:queryString&gt; &lt;/rdfs:Class&gt;</pre> <pre>&lt;rdfs:Class rdf:ID="QueryResult_ID1" slice:resultName=books slice:useAsSessionVar="Yes"&gt;   &lt;rdfs:subClassOf rdf:resource="#&amp;slice:QueryResult" /&gt; &lt;/rdfs:Class&gt;</pre> <pre>&lt;rdf:Property rdf:ID="result-ref_ID1"&gt;   &lt;rdfs:subPropertyOf rdf:resource="#&amp;slice#result- ref" /&gt;   &lt;rdfs:domain rdf:resource="#Query_ID1" /&gt;   &lt;rdfs:range rdf:resource="#QueryResult_ID1" /&gt; &lt;/rdf:Property&gt;</pre>
<p><b>Query Condition</b></p> <p>Esta query es una condición sobre la porción <i>Recommendation</i>. Comprueba que las condiciones especificadas en el requisito 2 (i.e. el grado de interés en el autor del libro debe ser mayor que 10). Si la condición se cumple, las instancias apropiadas de la porción <i>Recommendation</i> serán visibles.</p> <pre>&lt;rdfs:Class rdf:ID:"SliceCondition_ID1"&gt; &lt;rdfs:subclassOf rdf:resource="http://wwwis.win.tue.nl/~hera/ns/slice#SliceCo</pre>

```

ndition"/>
    <slice:queryString>
        SELECT B
        FROM
        {B}<cm:booksToAuthor>(BTA)<um:ID>{ID} ,
        <um:userToInterest>{UI}<um:degree>{D} ,
        <um:userToInterest>{UTI}<um:interestToAuthor>{ITA}<
um:ID>{ID2} ,
        {session}<session:user>{U}

        WHERE D>100 and ID= ID2 and B in Select * from
session:books
    </slice:queryString>
</rdfs:Class>

<rdf:Property rdf:ID="condition-ref_ID1">
    <rdfs:subPropertyOf rdf:resource="#Slice#condition-
ref"/>
    <rdfs:domain
rdf:resource="#SLice.Recommendation_ID1" />
    <rdfs:range rdf:resource="#SliceCondition_ID1" />
</rdf:Property>

```

## D.6 Conclusiones

### D.6.1 Contribuciones

El primer objetivo de esta tesis ha sido añadir personalización al método de diseño Object Oriented Hypermedia (OO-H). Con este propósito, OO-H se ha extendido llamándose A-OOH (adaptive OO-H), cuyo proceso de diseño está basado en el Proceso Unificado (UP) y no en el modelo en espiral como OO-H. El motivo de cambiar el proceso de diseño es el que los sistemas adaptivos necesitan un proceso de ingeniería del software adecuado para su desarrollo. Los modelos de OO-H también han sido modificados. El primer paso ha sido definir perfiles UML para cada uno de los modelos. Los modelos también han sido mejorados para el soporte de modelado de adaptividad y también se han añadido algunos modelos, concretamente un modelo de presentación y un modelo de personalización. El Capítulo 3 presenta los fundamentos de Adaptive OO-H (A-OOH) describiendo su proceso de diseño y los diferentes diagramas considerados. Como resultado de la ejecución de las actividades de cada una de las fases del proceso de diseño, en A-OOH obtenemos un conjunto de modelos, reflejando las vistas de la interfaz a generar

Las metodologías de diseño Web proveen una aproximación conceptual y sistemática para el diseño de aplicaciones Web (complejas) y su implementación. La mayoría de estas metodologías permiten desarrollar sitios Web donde la personalización está embebida en los modelos de diseño. Esto puede causar problemas como el difícil mantenimiento de la personalización y la inhabilidad de reutilizar una especificación de personalización entre diferentes aproximaciones. En el Capítulo 4 hemos presentado una solución basada en un lenguaje de reglas de alto nivel llamado PRML (Personalization Rules Modeling Language) que permite especificar la personalización en tiempo de diseño como un concepto ortogonal al sitio Web, independiente de la tecnología subyacente. El diseñador puede usar este lenguaje para especificar en tiempo de diseño la personalización a ejecutar en tiempo de ejecución (Capítulos 4 y 5).

Además el hecho de considerar la personalización como una disciplina completa, presente en diferentes campos además de la ingeniería Web nos sugiere la figura de un *diseñador* responsable de definir la personalización con conocimientos de psicología y sociología. En consecuencia, el *diseñador de personalización* no tiene porqué ser un programador con experiencia. Este diseñador puede encontrarse con el problema de tener diversas formas de especificar la personalización, tantas como el número de metodologías Web existentes que la soporta. Dependiendo de la metodología a utilizar, el *diseñador de personalización* debería aprender un Nuevo lenguaje para especificar la personalización. Esto es una tarea pesada para el diseñador, que como hemos comentado antes, no tiene porqué ser un programador experimentado. Nosotros proponemos que el diseñador utilice PRML para especificar la personalización y después esta especificación se transforma al lenguaje concreto del método de diseño Web utilizado (estas transformaciones se explican en los Capítulos 7 y 8). Esto reduciría la curva de aprendizaje del diseñador de personalización.

El doble propósito de PRML es hacer posible el reuso de estrategias de personalización entre diferentes aproximaciones y también permitir la definición de estrategias de personalización más complejas (y específicas). Para este propósito se han definido dos niveles de conformidad del lenguaje: PRML Lite y PRML Full.

PRML Full (explicado en el Capítulo 5) provee al diseñador un medio de especificar acciones de personalización más complejas que PRML Lite (presentado en el Capítulo 4) limitando el reuso. Las acciones de PRML Full son necesarias para especificar estrategias de personalización más avanzadas. Con este propósito PRML Full también permite reconocer acciones de navegación más complejas, necesarias para definir estrategias de personalización eficientes.

Otra contribución importante es AWAC, una herramienta CAWE prototipo para la generación automática de aplicaciones Web usando PRML (presentada en el Capítulo 6). Esta herramienta implementa la metodología A-OOH (Adaptive Object Oriented Hypermedia). El *input* de la herramienta AWAC es el conjunto de modelos de diseño A-OOH necesarios para modelar el sitio Web adaptivo a generar. El *output* es el sitio Web adaptivo generado con su base de datos. Una vez generado, el sitio Web adaptivo contiene tres módulos para el manejo de la personalización, que en tiempo de ejecución, analizan los eventos de navegación del usuario y adaptan el sitio Web de acuerdo a la(s) regla(s) de personalización lanzada(s). Las reglas de personalización se pueden editar de un modo independiente del resto de la aplicación, lo cual mejora el mantenimiento de la personalización.

En esta tesis argumentamos que PRML Lite se puede exportar a diferentes metodologías. Esto implica que el lenguaje PRML es reusable así como las estrategias definidas con él. Los Capítulos 7 y 8 han explicado como PRML Lite puede ser exportado a las aproximaciones UWE y Hera definiendo un conjunto de transformaciones para mostrarlo de manera conceptual. Estas transformaciones se definen usando el lenguaje QVT. El apéndice C de esta tesis describe un prototipo (PRML translator) que automáticamente implementa las transformaciones definidas.

### **D.6.2 Limitaciones**

Teniendo en cuenta que es imposible cubrir todos los aspectos de la personalización, se impusieron ciertos límites a este trabajo y también existen algunas limitaciones:

- *Adaptación de la presentación.* No se considera la personalización de la presentación en el presente trabajo.
- *Limitación considerando los sitios web dependientes de dispositivo.* Este trabajo no considera en detalle la adaptación basada en contexto, ya que está fuera del ámbito de la presente tesis.
- *PRML FULL parcialmente implementado.* No se ha implementado toda la funcionalidad de PRML en la herramienta AWAC. PRML Lite está implementado completamente pero no ha sido posible implementar algunas acciones de PRML Full como agrupar usuarios, añadir o borrar enlaces dinámicamente, soporte de eventos complejos y reglas de comportamiento.
- *Herramienta AWAC: no editor gráfico.* La herramienta AWAC no dispone de un editor gráfico para dibujar los modelos de A-OOH.
- *Limitaciones de la portabilidad de PRML.* Definir transformaciones de PRML a otras aproximaciones (además de UWE y Hera) reforzaría los resultados obtenidos. Sin embargo, debe tenerse en cuenta que las transformaciones realizadas en esta disertación son suficientes para alcanzar nuestros propósitos.
- *Limitaciones del experimento.* El número de visitantes al sitio web generado puede considerarse pequeño, debido al tópico del caso de estudio. Sería interesante realizar un sitio web más grande con un mayor número de visitantes.

### D.6.3 Trabajo Futuro

Como trabajo futuro, además de implementar la funcionalidad de PRML por complete, nos gustaría añadir una interfaz gráfica para poder definir los modelos utilizando AWAC. Ahora, para definir los modelos A-OOH y generar los ficheros XMI utilizamos la herramienta Enterprise Architect Design tool [Enterprise Architect] en la que hemos definido los perfiles UML necesarios para el modelado de los diagramas de A-OOH.

Otra acción futura sería extender PRML Full para soportar acciones de adaptación en relación a la presentación del sitio web.

También planeamos realizar transformaciones de PRML a otras aproximaciones Web como WebML u OOHDM para mejorar la reusabilidad del lenguaje.

Otra tarea a realizar sería modelar experimentos más extensos para corroborar los resultados obtenidos en esta disertación. Nos gustaría evaluar explícitamente la satisfacción del usuario por medio de cuestionarios para confirmar los resultados obtenidos en este trabajo.



Universitat d'Alacant  
Universidad de Alicante

## References

1. Agosti, M., Melucci M. and Crestain F.: (1995) *Automatic authoring and construction of hypermedia for information retrieval*. *Multimedia Systems* 3 (1), 15-24.
2. amazon, <http://www.amazon.com>
3. ANTLR, ANother Tool for Language Recognition, <http://www.antlr.org/>
4. Albrecht F., Koch N. & Tiller T. (1999). Making Web-based training more effective. Proceedings of the Seventh Workshop ABIS-99: Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen, T. Joerding (Ed.).
5. Apple W.P. Fok & Horace H.S. Ip, (2004) *Personalized education—An exploratory study of learning pedagogies in relation to personalization technologies*, Lecture Notes in Computer Science 3143, Advances in Web-based Learning (ICWL 2004), (New York: Springer Verlag, 2004), 407–415.
6. Atzeni, P., Mecca, G. and Merialdo, P.: (1998) Design and Maintenance of Data-Intensive Web Sites. Pages 436-449 of Advances in Database Technology – EDBT'98.
7. Backus, J. W., Wegstein, J. H., van Wijngaarden, A., Woodger, M., Bauer, F. L., Green, J., Katz, C, McCarthy, J., Perlis, A. J., Rutishauser, H., Samelson, K. and Vauquois, B. (1960). *Revised Report on the Algorithmic Language ALGOL 60*. In Communications of the ACM, Vol. 3 No.5, pages 299-314, ACM Press, ISSN 0001-0782
8. Ballacker K., Lawrence S, & Giles L. (2000). Discovering Relevant Scientific Literature on the Web. *IEEE Intelligent Systems*, 15 (2), 42-47.
9. Baresi L., Garzotto F., Paolini P.: (2001) *Extending UML for Modelling Web Applications*. Annual Hawaii Int. Conf. on System Sciences. pp. 1285 - 1294. Miami, USA. January, 2001.

10. Bevan, N. (2005) *Guidelines and Standards for Web Usability*. Proceedings of HCI International. Lawrence Erlbaum.
11. Bia, A., Garrigós, I. and Gómez, J. (2004) *A Newsletters Service based on XML-TEI*. ALLC/ACH 2004. Computing and Multilingual, Multicultural Heritage. The 16th Joint International Conference of the Association for Literary and Linguistic Computing and the Association for Computers and the Humanities. Göteborg University, Sweden. Conference Proceedings, Pages 19-22.
12. Bia, A., Garrigós, I. and Gómez, J. (2004). *Personalizing Digital Libraries at Design Time: The Miguel de Cervantes Digital Library Case Study*. Fourth International Conference on Web Engineering (ICWE'04). July 2004 Munich, Germany. ISSN: 0302-9743 LNCS 3140 Pags 225-229. Springer-Verlag
13. Blom J. (2000) *Personalization – A Taxonomy*, ACM 2000. ISBN:1-58113-248-4.
14. Boyle, T. and Encarnacion A.O.: (1994) *MetaDoc: an Adaptive Hypertext Reading System*. User models and User Adapted Interaction 4(1), 1-19.
15. Brickley, D., Guha, R.: (2004) *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C Recommendation 10 February 2004
16. Brusilovsky P. (1997). *Efficient Techniques for Adaptive Hypermedia*. Intelligent Hypertext: Advanced Techniques for the World Wide Web. Nicholas C. & Mayfield J. (Eds.), Springer Verlag, 12-30.
17. Brusilovsky, P., Eklund, J., and Schwarz, E. (1998) *Web-based education for all: A tool for developing adaptive courseware*. Computer Networks and ISDN Systems (Proceedings of Seventh International World Wide Web Conference), Vol. 30 (1-7), pp. 291-300, 1998.
18. Brusilovsky, P. (2001). *Adaptive Hypermedia*. In User Modeling and User Adapted Interaction, 11 (1/2), pages 87-110, Kluwer Academic Publishers
19. Brusilovsky, P. (1996). *Methods and techniques of adaptive hypermedia*. In User Modeling and User-Adapted Interaction, 6 (2-3), pages 87-129, Springer Science+Business Media B.V, ISSN 0924-1868

20. Brusilovsky P., Schwarz E. and Weber G.: (1996a). *ELM-ART: An Intelligent Tutoring System on World Wide Web*. Proceeding of Third International Conference on Intelligent Tutoring Systems ITS-96, LNCS 1086, Springer Verlag, 261-269.
21. Brusilovsky P., Schwarz E. and Weber G.: (1996b) *A Tool for Developing Adaptive Electronic Textbooks on WWW*. Proceeding of WebNet '96, World Conference of the Web Society, 64-69.
22. Bulterman, D. C. A., Rutledge, L., Hardman, L. & van Ossenbruggen, J.: (1999) *Supporting Adaptive and Adaptable Hypermedia Presentation Semantics*. The 8<sup>th</sup> IFIP 2.6 Working Conference on Database Semantics (DS-8): Semantic Issues in Multimedia Systems.
23. Cachero, C., Garrigós, I. and Gómez, J. (2002) *Personalización de Aplicaciones en OO-H*. Quintas Jornadas Iberoamericanas de Ingeniería de Requisitos y Ambientes Software (IDEAS'02). La Habana, Cuba. Abril 2002. Actas de la Conferencia.
24. Cailiau, R. and Ashman, H. (1999). Hypertext in the Web — a history. In *ACM Computing Surveys (CSUR), Volume 31, Issue 4es*, Article No. 35, ACM Press, ISSN 0360-0300
25. Campbell, B. and Goodman, J.M.: (1998) *HAM: A General Purpose Hypertext Abstract Machine*. Communications of the ACM, 31(7): 856-861.
26. Casteleyn, S., De Troyer, O.: (2001) *Structuring Web Sites Using Audience Class Hierarchies*, In Conceptual Modeling for New Information Systems Technologies, ER 2001 Workshops, HUMACS, DASWIS, ECOMO, and DAMA, Lecture Notes in Computer Science , Vol. 2465, Publ. Springer-Verlag, ISBN 3-540-44-122-0, Yokohama, Japan.
27. Casteleyn, S., De Troyer, O., Brockmans, S.: (2003) *Design Time Support for Adaptive Behaviour in Web Sites*, In Proc. of the 18th ACM Symposium on Applied Computing, Melbourne, USA, pp. 1222 – 1228.
28. Casteleyn, S. ,Garrigós, I. ,Plessers, P.: (2004) *Pattern Definition to Refine Navigation Structure in Hypermedia/Web Applications*, In Proceedings of the IADIS International Conference WWW/Internet 2004 (ICWI2004).
29. Casteleyn, S., Garrigós, I., De Troyer, O.: (2005) *Automatic Runtime Validation and Correction of the Navigational Design of Web Sites*, In Web

- Technologies Research and Development - APWeb 2005, pp. 453 - 463, Eds. Yanchun Zhang, Katsumi Tanaka, Jeffrey Xu Yu, Shan Wang, Minglu Li, Publ. Springer-Verlag, ISBN 3-540-25207-X, Shanghai, China
30. Casteleyn, S.: (2005) *Designer Specified Self Re-organizing Websites*, Phd thesis, Vrije Universiteit Brussel
  31. Ceri, S., Daniel, F., Matera, M. and Facca, F. (2007) *Model-driven - Development of Context Aware Web Applications*. ACM Transactions on Internet Technology (TOIT), Volume 7, Number 1, ISSN 1533-5399, February 2007.
  32. Ceri, S., Daniel, F. and Matera, M. (2003) *Extending WebML for Modeling Multi-Channel Context-Aware Web Applications*. Proc. of the WISE - MMIS'03 Workshop (Mobile Multi-channel Information Systems), Roma, December 2003, IEEE Press.
  33. Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S. and Matera, M.(2002). *Designing Data-Intensive Web Applications*. Morgan Kaufmann Publishers Inc., ISBN 1558608435
  34. Ceri, S., Fraternali, P., Bongio, A. and Maurino, A. (2000). Modeling data entry and operations in WebML. In *Lecture Notes In Computer Science; Vol. 1997, Selected papers from the Third International Workshop WebDB 2000 on The World Wide Web and Databases*, pages 201-214, Springer-Verlag, ISBN 3-540-41826-1
  35. Chakravarthy, S., Le, R., and Dasari, R. (1999). *ECA Rule Processing in Distributed and Heterogeneous Environments*. In Proceedings of the international Symposium on Distributed Objects and Applications. September 05 - 07. DOA. IEEE Computer Society, Washington, DC, 330.
  36. Cingil I., Dogac A., & Azgin A. (2000) *A broader approach to personalization*, Communications of the ACM, Vol.43, No. 8.
  37. Cowan, D. and Lucena, C., "Abstract Data Views, An Interface Specification Concept to Enhance Design for Reuse", IEEE Transactions on Software Engineering, Vol.21, No.3, March 1995.
  38. Chorianopoulos, K. and Spinellis, D. (2002) *A metaphor for personalized television programming*. In Proceedings of the 7th ERCIM Workshop on User Interfaces for All, Paris (Chantilly), France, October. Springer-Verlag.

39. Daniel, F., Matera, M., Morandi, A., Mortari, M. and Pozzi, G.(2007) *Active Rules for Runtime Adaptivity Management*. Second international workshop on Adaptation and Evolution in Web Systems Engineering (AEWSE'07). In Workshop proceedings on the seventh International conference of Web Engineering. Como, Italy, pp 28-42.
40. Daniel, F., Matera, M., Pozzi, G.: (2006) *Combining Conceptual Modeling and Active Rules for the Design of Adaptive Web Applications*. First international workshop on Adaptation and Evolution in Web Systems Engineering (AEWSE'06). In Workshop proceedings of the sixth International conference of Web Engineering. ISBN: 1-59593-435-9 ACM Digital Library.
41. Dayal, U. (1988). *Active Database management systems*. In Proceedings of the Third International Conference on Data and Knowledge Bases, pages 150, 169, Publ. Morgan Kaufmann, ISBN 0-93413-95-8
42. Db4o, Database For Objects [www.db4o.com](http://www.db4o.com)
43. De Bra, P.: (1999). *Design Issues in Adaptive Hypermedia Application Development*. In Proceedings of the Second Workshop on Adaptive Systems and User Modeling on the *World Wide Web*, pages 29-39
44. De Bra, P., Houben, G.J. and Wu, H.: (1999). *AHAM: A Dexter-based Reference Model for Adaptive Hypermedia*. In Proceedings of the ACM Conference on Hypertext and Hypermedia, pages 147-156, ACM Press, ISBN 1-58113-064-3
45. De Bra, P., Houben, G.J. and Kornatzky, Y. (1992). An Extensible Data Model for Hyperdocuments. In *4th ACM Conference on Hypertext*, pages 222-231, ACM Press, ISBN 0-89791-547-X
46. De Bra, P. and Calvi, L.: (1997) *Creating adaptive hyperdocuments for and on the web*. In Proc. of World Conference of the WWW, Internet, and Intranet (WebNet'97), Toronto, Canada, pages 149-155.
47. De Bra P. and Calvi L.: (1998a). *AHA: A Generic Adaptive Hypermedia System*. Proceedings of the 2nd Workshop on Adaptive Hypertext and Hypermedia, HYPERTEXT'98, Pittsburgh, PA, USA, pages 20-24.
48. De Bra P. and Calvi L.: (1998b) *AHA! an open adaptive hypermedia architecture*. The New Review of Hypermedia and Multimedia, 4:115-139.

49. De Bra, P., Stash, N., De Lange, B. (2003) *AHA! Adding Adaptive Behavior to Websites*. Proceedings of the NLUUG Conference, pp. n-n+10, Ede, The Netherlands, May
50. De Troyer, O.: (2005) *Audience-driven web design*, In Encyclopedia of Information Science and Technology, pp. 184 - 187, Eds. Mehdi Khosrow-Pour, Publ. IDEA Group Publishing, ISBN 1-59140-553-X.
51. De Troyer, O., Casteleyn, S.: (2001) *The Conference Review System with WSDM*, In First International Workshop on Web-Oriented Software Technology, IWWOST'01, (also <http://www.dsic.upv.es/~west2001/iwwost01/>), Eds. Oscar Pastor, Valencia (University of Technology), Spain.
52. De Troyer, O., Leune, C.: (1998) *WSDM: A User-Centered Design Method for Web Sites*, In Computer Networks and ISDN systems, Proceedings of the 7th International World Wide Web Conference, pp. 85 - 94, Publ. Elsevier, Brisbane, Australia
53. De Troyer, O., Casteleyn, S.: (2004) *Designing Localized Web Sites*, In Proceedings of the 5th International Conference on Web Information Systems Engineering (WISE2004), pp. 547 - 558, Eds. Zhou, X., Su, S., Papazoglou, M.P., Orłowska, M.E., Jeffery, K.G., Publ. Springer-Verlag, ISBN 3-540-23894-8, Brisbane, Australia
54. Encarnação M. (1997). *Concept and Realization of Intelligent Support in Interactive Graphics Applications*. Ph.D. Thesis.
55. Enterprise Architect - UML Design Tool, <http://www.sparxsystems.com>
56. Facca F. M., Ceri S., Armani J. and Demaldé V. (2005), *Building Reactive Web Applications*. Poster at WWW2005, Chiba, Japan.
57. Fiala, Z., Frasincar, F., Hinz, M., Houben, G.J., Barna, P. and Meissner, K.(2004). *Engineering the presentation layer of adaptable web information systems*. In Web Engineering 4th International Conference, ICWE 2004, volume 3140 of Lecture Notes in Computer Science, pages 459-472, Springer, ISBN 3-540-22511-0
58. Fink J., Kobsa A. & Nill A. (1997). *Adaptable and Adaptive Information Access for all Users Including the Disabled and the Elderly*. User Modeling

- Proceedings of the Sixth International Conference, UM97, Jameson A., Paris C. and Tasso C. (Eds.), Springer Verlag Wien, 171-173.
59. Fraternali, P. (2000) *Tools and Approaches for Developing Data-Intensive Web Applications: a survey*. ACM Computing Surveys.
  60. Frasincar, F., Barna, P., Houben, G.-J. and Fiala Z. (2004). Adaptation and reuse in designing web information systems. In *International Conference on Information Technology: Coding and Computing (ITCC'04)*, pages 387-291, IEEE Computer Society
  61. Frasincar, F. and Houben, G.-J. (2002). Hypermedia presentation adaptation on the semantic web. In *Adaptive Hypermedia and Adaptive Web-Based Systems, Second International Conference, AH 2002, volume 2347 of Lecture Notes in Computer Science*, pages 133-142. Springer, ISBN 3-540-43737-1
  62. Frasincar, F., Houben G.J., Barna P. (2006) *HPG: The Hera Presentation Generator*. Journal of Web Engineering, Vol. 5, No. 2, p. 175-200, Rinton Press
  63. Furuta, R., Stotts, P.D.: (1990) *The Trellis hypertext reference model*. In Proc. of NIST Hypertext Standardization Workshop, Gaithersburg, MD, USA, pages 83-93.
  64. Gamma E., Helm R., Johnson R. & Vlissides J. (1995). *Design Patterns. Elements of REusable Object-Oriented Software*. Addison Wesley.
  65. Garlatti S., Iksal S. & Kervella P. (1999). *Adaptive On-line Information System by Means of a Task Model and Spatial Views*. Second Workshop on Adaptive Systems and User Modeling on the WWW, Toronto, Canada, pp:59-66.
  66. Garrigós, I., Cachero, C. and Gómez, J. (2002) *Modelado Conceptual de Aplicaciones adaptivas y proactivas en OO-H*. II Taller sobre Ingeniería del Software Orientada al Web (Web Engineering) (JISBD'02). El Escorial, Madrid Noviembre.
  67. Garrigós, I., Gómez, J. and Cachero, C.: (2003) *Modeling Dynamic Personalization in Web Applications*, Third International Conference on Web Engineering (ICWE), LNCS 2722, pp 472-475. Springer-Verlag.

68. Garrigós, I., Gómez, J. and C. Cachero. (2003) *Modelling Adaptive Web Applications*. International WWW/Internet 2003 Conference (ICWI'03). November 2003 Algarve, Portugal.
69. Garrigós, I., Gómez, J. and C. Cachero. (2003) *Tratamiento de la Personalización Dinámica en Modelos Conceptuales de Aplicaciones Web*. VIII Jornadas de Ingeniería del Software y Bases de Datos. Noviembre 2003, Alicante.
70. Garrigós, I., Casteleyn, S., Gómez, J. (2005) *A Structured Approach to Personalize Websites using the OO-H Personalization Framework*. Seventh Asia Pacific Web Conference (APWEB'05). March-April 2005, Shangai China. ISSN: 0302-9743 LNCS 3399. Pags 695-705. Springer-Verlag
71. Garrigós I., Gómez J., Barna P., Houben G.J.: (2005) *A Reusable Personalization Model in Web Application Design*. International Workshop on Web Information Systems Modeling (WISM 2005) July Sydney, Australia.
72. Garrigós I. and Gómez J.:(2006) *Modeling User Behaviour Aware Websites with PRML*. International Workshop on Web Information Systems Modeling (WISM 2006) June 6, Luxembourg, Grand Duchy of Luxembourg.
73. Garrigós I., Cruz C. and Gómez J. (2007) *A Prototype Tool for the Automatic Generation of Adaptive Websites* Proc. of the 2nd International Workshop on Adaptation and Evolution in Web Systems Engineering (AEWSE'07), Como, Italy, July 2007. CEUR Workshop Proceedings, ISSN 1613-0073. <http://CEUR-WS.org/Vol-267/paper2.pdf>
74. Garrigós I. and Gómez J. (2007): *Modelado de Aplicaciones Web Reactivas al Usuario*. XII Jornadas de Ingeniería del Software y Bases de Datos. ISBN: 978-84-9732-595-0. Septiembre 2007, Zaragoza.
75. Garzotto, F., Paolini, P., and Schwabe, D.:(1991) *HDM - A Model for the Design of Hypertext Applications*. Proceedings of Hypertext '91, ACM Press,December 1991.
76. Garzotto F., Paolini P & Schwabe D. (1993). *HDM: A Model-based Approach to Hypertext Application Design*. ACM Transactions of Information Systems, 11(1), 1-26

77. Gómez, J., Cachero, C. and Pastor, O. (2000). *Extending an Object-Oriented Conceptual Modeling Approach to Web Application Design*. In Proceedings of the 12th International Conference on Advanced Information Systems Engineering - LNCS 1789, pages 79-93, Springer-Verlag, ISBN:3-540-67630-9
78. Gómez, J., Cachero, C. and Pastor, O. (2001). *Conceptual Modeling of Device-Independent Web Applications*. In IEEE Multimedia Special Issue on Web Engineering, pages 26-39, IEEE Computer Society Press, ISSN 1070-986X
79. Gray, J. et al. (2003): *An Approach for Supporting Aspect-Oriented Domain Modeling*. Generative Programming and Component Engineering Second International Conference, GPCE 2003, Erfurt, Germany, September 22-25, 2003, Proceedings Volume 2830/2003 978-3-540-20102-1 GPCE 2003, pp 151-158
80. Halasz, F. and Schwartz, M. (1990). The Dexter Reference Model. In *Proceedings of the NIST Hypertext Standardization Workshop*, pages 95-133
81. Halasz, F. and Schwartz, M., (1994). The Dexter Reference Model. In *Communications of the ACM, Vol 37, issue 2 (February)*, pages 30- 39, ACM Press, ISSN 0001-0782
82. Halpin, T. (1995), *Conceptual Schema and Relational Database Design*, Second Edition, Prentice Hall Australia.
83. Hardman, L., Bulterman, D.C.A. and van Rossum G.: (1994) *The Amsterdam hypermedia model: adding time and context to the Dexter model*. In Communications of the ACM Volume 3 , Issue 2, pages 50 - 62, ACM Press, ISSN 0001-0782
84. Helmes, L., Razum, M. and Barth, A.:(1995) *Concept of a hypertext interface for the information retrieval in complex factual databases*. In R.Kuhlen and M. Ritterberg (eds.): Hypertext- Information Retrieval-Multimedia. Konstanz University, pp. 175- 189.
85. Heimrich T. and Specht G. (2003) *Enhancing ECA Rules for Distributed Active Database Systems*. In Revised Papers from the Node 2002 Web and Database-Related Workshops on Web, Web-Services, and Database Systems. London, UK, October 07-10 pp.~199-205. Springer-Verlag, Berlin.

86. Hjelm, J., Martin, B. and King, P. (1998). WAP Forum – W3C Cooperation White Paper. <http://www.w3.org/TR/NOTE-WAP>
87. Höök K. (1998). Evaluating the Utility and Usability of an Adaptive Hypermedia System. *Knowledge-Based Systems*, Elsevier, 10, 311-319.
88. Houben, G.-J., Frasincar, F., Barna, P. and Vdovjak, R. (2004). Engineering the presentation layer of adaptable web information systems. In *Web Engineering 4th International Conference, ICWE 2004, volume 3140 of Lecture Notes in Computer Science*, pages 60-73, Springer, ISBN 3-540-22511-0.
89. Gronbaek, K.; Trigg, R. H.: (1996) *Toward a Dexter-based model for open hypermedia: unifying embedded references and link objects*. Proceedings of Hypertext'96; 1996 March 10-14; Washington, DC. New York:ACM; 149-160.
90. Jacobson, I., Booch, G. & Rumbaugh, J. (1999). *The Unified Software Development Process*. Reading, MA: Addison-Wesley. ISBN 0201571692
91. Kappel, G., Retschitzegger, W., Poll, W., & Schwinger, W.: (2001) *Modeling Ubiquitous Web Applications - The WUML Approach*. In: Proceedings of the International Workshop on Data Semantics in Web Information Systems (DASWIS 2001) Yokohama, Japan, 2001.
92. Kappel G., Retschitzegger W., Kimmerstorfer E., Proll B, Schwinger W. & Hofer Th. (2002) *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.
93. Kim, D.-W.: (1995) *WING-MIT: Das auf einer multimedialen und intelligenten Benutzerschnittstelle basierende tutorielle Hilfesystem für das Werkstoffinformationssystem WING-M2*. Workshop Adaptivität und Benutzermodellierung in interaktiven Systemen (ABIS 95), München.
94. Kim W. (2002) *Personalization: Definition, Status, and Challenges Ahead*, Published by ETH Zurich, Chair of Software Engineering JOT, ©2002, Vol. 1, No. 1.

95. Knapp A., Koch N., Moser F., Zhang, G.: ArgoUWE (2003) A CASE Tool for Web Applications. EMSISE03, 14 pages, online publication at <http://www.pst.informatik.unimuenchen.de/~kochn>
96. Kobsa, A., Koenemann, J. and Pohl, W. (2001). Personalised hypermedia presentation techniques for improving online customer relationships Source. In *The Knowledge Engineering Review archive Volume 16 , Issue 2 (March 2001)*, pages 111 – 155, Cambridge University Press, ISSN: 0269-8889
97. Koch, N. (2000) *Hypermedia Systems Development based on the Unified Process*. Tech. rept. Ludwig-Maximilians- Universität Munchen.
98. Koch, N. (2001). *Software Engineering for Adaptive Hypermedia Systems, Reference Model, Modeling Techniques and Development Process*, PhD Thesis, Verlag UNI-DRUCK, ISBN 3-87821-318-2
99. Koch, N., Kraus, A. and Hennicker, R. (2001). *The Authoring Process of the UML-based Web Engineering Approach*. In Proceedings of the 1st International Workshop on Web-Oriented Software Technology.
100. Koch, N. and Kraus, A. (2002) *The Expressive Power of UML-based Web Engineering*, In Proc. of the 2nd. Int. Workshop on Web-Oriented Software Technology, CYTED, Málaga, Spain, 105-119, June
101. Kruchten, Philippe (2004). *The Rational Unified Process: An Introduction* (3rd Ed.) ISBN 0-321-19770-4
102. Lange, D. (1990): *A Formal Approach to Hypertext using Post-Prototype Formal Specification*. VDM Europe 1990. pp. 99-121
103. Mecca, G., Merialdo, P., Atzeni, P. and Crescenzi, V.: (1999) *The ARANEUS Guide to Web Site Development*. Technical Report University of Rome.
104. Mernik, M., Heering, J., Sloane, A. (2005) *When and how to develop domain-specific languages*, ACM Computing Surveys (CSUR), v.37 n.4, p.316-344, December.
105. Nielsen, J. (1992): *Finding usability problems through heuristic evaluation*. In Proc. of the SIGCHI Conf on Human factors in computing systems. Monterey, California, United States pp: 373 – 380.

106. OMG, Object Management Group: Meta Object Facility (MOF) v1.4, OMG doc.formal/02-04-03
107. OMG MDA, Object Management Group: MDA Guide 1.0.1. <http://www.omg.org/cgi-bin/doc?omg/03-06-01> (Visited January 2006)
108. OMG QVT, Object Management Group: MOF 2.0 Query/View/Transformation. <http://www.omg.org/cgi-bin/doc?ptc/2005-11-01> (Visited January 2006)
109. OpenRDF, The SeRQL query language, rev. 1.1, url: <http://www.openrdf.org/doc/users/ch06.html>.
110. Paterno, F., Mancini, C. and Meniconi, S.: (1997) *ConcurTaskTrees: a Diagrammatic Notation for Specifying Task Models*, In Proceedings of INTERACT 97, Chapman & Hall, 362-366
111. Resource Description Framework RDF(S), <http://www.w3.org/RDF/>
112. Rich, E., (1983) *Users are individuals: individualizing user models*. International Journal of Man-Machine Studies, 18, pp. 199-214.
113. Rossi, G., Schwabe, D. and Guimarães, R. (2001). *Designing personalized web applications*. In WWW2001, pages 275-284, Hong-Kong, May, ISBN 1-58113-348-0
114. SALTForum.Org (2005). Speech Application Language Tags (SALT). <http://saltforum.org/>
115. Schwabe, D. and Moura, S. (2003). Interface development for hypermedia applications in the semantic web. In *Proceedings of the WebMedia & LA-Web 2004 Joint Conference 10th Brazilian Symposium on Multimedia and the Web 2nd Latin American Web Congress - Volume 00*, pages 106-113, IEEE Computer Society, ISBN 0-7695-2237-8
116. Schwabe, D. and Rossi, G. (1995). The Object-Oriented Hypermedia Design Model. In *Communications of the ACM 38(8)*, pages 45-46, ACM Press, ISSN 0001-0782
117. Schwabe, D. and Rossi, G. (1998). An object oriented approach to web-based applications design. In *Theory and Practice of Object Systems, 4(4)*, pages 207-225, John Wiley & Sons, Inc , ISSN 1074-3227

118. Schwabe, D. and Rossi, G. (2001) *A Conference Review System with OOADM*. In First International Workshop on Web-Oriented Software Technology, 05 2001.
119. Schwabe, G. Szundy, de S. Moura, and F. Lima (2004). *Design and implementation of semantic web applications*. In WWW 2004 Workshop on Application Design, Development and Implementation Issues in the Semantic Web
120. Spivey J. (1992). *The Z Notation: A Reference Manual*. Series in Computer Science. Prentice Hall International, second edition.
121. The Personalized Medicine Coalition  
<http://www.personalizedmedicinecoalition.org/>
122. Thomas, C. G.: (1995) *Basar: A framework for integrating agents in the World Wide Web*. IEEE Computer 28 (5), 84-86.
123. Thomas, C.G. and Fischer G.: (1996) *Using agents to improve the usability and usefulness of the World Wide Web*. Fifth International Conference on User Modeling (UM-96), Kailua-Kona, Hawaii, pp. 5-12.
124. UML. 1999 (June). OMG Unified Modelling Language Specification. <http://www.rational.com/uml/>
125. UML 2.0 Unified Modeling Language. The Object Management Group. <http://www.uml.org/#UML2.0>
126. UML 2.0 OCL specification [www.omg.org/docs/ptc/03-10-14.pdf](http://www.omg.org/docs/ptc/03-10-14.pdf)
127. Van Deursen, A., Klint, P., and Visser, J., Domain-Specific Languages: An Annotated Bibliography. ACM SIGPLAN Notices, 2000. 35(6): p. 26-36.
128. Van Ossenbruggen J. and Eliëns A. (1995). The Dexter Hypertext Reference Model in Object-Z. <http://www.cs.vu.nl/~dejavu/papers/dexter-full.ps.gz>
129. Vassileva, J.:(1996) *A task-centered approach for user modelling in a hypermedia office documentation system*. User Models and User Adapted Interaction 6.

130. Vdovjak, R., Frasincar, F., Houben, G.-J. and Barna, P. (2003). *Engineering semantic web information systems in hera*. In Journal of Web Engineering, 2(12), pages 3-26, Rinton Press, ISSN 1540-9589.
131. Wang J., & Lin J. (1998) *Are personalization systems really personal? – Effects of conformity in reducing information overload*, Proceedings of the 36th Hawaii International Conference on Systems Sciences (HICSS'03), 0-7695-1874-5/03, © 2002 IEEE., 2002Warmer J., Kepple A. The Object Constraint Language Precise Modeling with UML. Addison-Wesley, 1998
132. Waterworth, J.A.: (1996). A pattern of islands: exploring public information space in a private vehicle. In: P. Brusilovsky, P. Kommers and N. Streitz (eds.): Multimedia, Hypermedia and Virtual Reality: Models, Systems, and Applications. Lecture Notes in Computer Science, Berlin: Springer-Verlag, pp. 266-279.
133. Web Ontology Language, <http://www.w3.org/TR/owl-features>
134. WebRatio website, <http://www.webratio.com>
135. Wu, H. (2002) *A Reference Architecture for Adaptive Hypermedia Applications*, PhD thesis, Eindhoven University of Technology, The Netherlands, November, ISBN 90-386-0572-2. <http://www.wis.win.tue.nl/ah/thesis/wu.pdf>
136. W3C (2004) Voice Extensible Markup Language (VoiceXML) Version 2.0 <http://www.w3.org/TR/voicexml20/>
137. XML Metadata Interchange  
[www.omg.org/technology/documents/formal/xmi.htm](http://www.omg.org/technology/documents/formal/xmi.htm)
138. Ying Q., Kang Z., HongAn W. and Xiang Li.. (2007) Developing event-condition-action rules in real-time active database, SAC '07: Proceedings of the 2007 ACM symposium on Applied computing, Seoul, Korea, March 11-15 pp.~511-516. ACM, New York, NY, USA.



Universitat d'Alacant  
Universidad de Alicante

Reunido el Tribunal que suscribe en el día de la fecha acordó otorgar, por \_\_\_\_\_ a la Tesis Doctoral de Don/Dña. Irene Garrigós Fernández la calificación de \_\_\_\_\_ .

Alicante \_\_\_\_\_ de \_\_\_\_\_ de \_\_\_\_\_

El Secretario,

El Presidente,



Universitat d'Alacant  
Universidad de Alicante  
**UNIVERSIDAD DE ALICANTE**  
**Comisión de Doctorado**

La presente Tesis de D. \_\_\_\_\_ ha sido registrada con el nº \_\_\_\_\_ del registro de entrada correspondiente.

Alicante \_\_\_\_ de \_\_\_\_\_ de \_\_\_\_\_

El Encargado del Registro,

La defensa de la tesis doctoral realizada por D/D<sup>a</sup> Irene Garrigós Fernández se ha realizado en las siguientes lenguas:                    y                    , lo que unido al cumplimiento del resto de requisitos establecidos en la Normativa propia de la UA le otorga la mención de “Doctor Europeo”.

Alicante,                    de                    de

EL SECRETARIO

EL PRESIDENTE



Universitat d'Alacant  
Universidad de Alicante