A Prototype Tool for the Automatic Generation of Adaptive Websites

Irene Garrigós[1], Cristian Cruz[1] and Jaime Gómez[1]

[1] Universidad de Alicante, IWAD, Campus de San Vicente del Raspeig, Apartado 99
03080 Alicante, Spain
{igarrigos, ccruz, jgomez}@dlsi.ua.es

Abstract. This paper presents AWAC, a prototype CAWE tool for the automatic generation of adaptive Web applications based on the A-OOH methodology. A-OOH (*Adaptive OO-H*) 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 of the adaptive Website to generate as an input. 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. These personalization rules are specified in an independent file so they can be updated without modifying the rest of the application logic.

1 Introduction

The continuous evolution of the WWW is reflected in the growing complexity of the Web applications with rapidly changing information and functionality. This evolution has lead to user disorientation and comprehension problems, as well as development and maintenance problems for designers. 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 [3,4,11,12,14,16], 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. In order to better tailor the site to one particular user, or a group of users, several Web methodologies provide (to some extent) personalization or adaptation support. Yet, few of these approaches provide an underlying CAWE[1] tool for Web Engineering and even less provide a tool to support personalization modeling (see next section for an overview). The lack of

[1] Computed Aided Web Engineering

such tools causes the personalization to be implemented in an ad-hoc manner. Moreover, the different (adaptive) methodologies are not properly tested.

In this paper, we present the fundamentals of the AWAC (Adaptive Web Applications Creator) prototype tool. This is a CAWE tool which automatically generates adaptive Web applications based on the A-OOH methodology developed at the University of Alicante (Spain). 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 allow the adaptivity at runtime of the final Web pages. The paper is structured as follows. In the next section, a study of the existing methodologies with a CAWE tool supporting adaptivity is presented. In Section 3 a brief introduction to the A-OOH method is given. The paper continues in Section 4 describing the AWAC architecture and some of the technologies used. This section also describes the personalization support in AWAC. Along the Sections 5, 6, and 7 the different steps for creating and running an adaptive Website using the AWAC tool are explained. A running example (online library) is used to describe the tool support. Finally, Section 8 sketches some conclusions and further work.

2 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) [9], 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[15] 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! [7] (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 [17] 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). To validate WSDM, at the University of Brussels (Belgium), a prototype tool was created for the support of the methodology [2]. It does not support personalization, but adaptivity for all the users. ArgoUWE [13] is the tool developed to support the UWE approach at the Luwdig Maximilians University of Munich (Germany). UWE supports personalization however it is not yet incorporated on the ArgoUWE tool.

3 A-OOH Fundamentals

The Adaptive OO-H method (A-OOH) is an extension of the OO-H (Object Oriented Hypermedia) approach [11] to support the modeling of adaptive (and personalized) Web applications. It supports most of the OO-H basic features and some updates and extensions for the support of adaptive Web sites modeling.
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. 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.

The set of A-OOH models are defined for the running example in Sections 5 and 6. 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 [5] 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 Web designers2 to define all the rules required to implement a personalization strategy. This language is called PRML (Personalization Rules Modelling Language) and will be shortly explained in Section 4.2. Next, the fundamentals of AWAC are explained.

4 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*[3](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 design time two new models are added (to the set of models): 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. 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 [18] format). The reason for choosing an XMI representation of the models is that this format can be easily generated from UML models (most UML tools allow this transformation). 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 is:

- **The generated adaptive Website (Web pages):** the actual 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 transitioning complexity 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 [6], the most

[2] Web designers are not necessarily experienced Web programmers.
[3] Sometimes called Conceptual Model

popular database for objects of open code, native to Java and .NET. Db4o eliminates the OO-versus-performance tradeoff: it allows you to store even the most complex object structures with ease, while achieving highest level of performance.

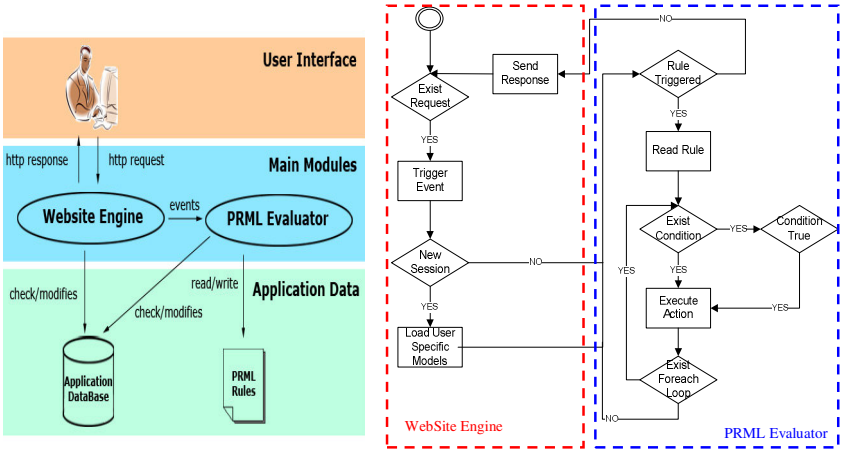

Fig. 1. (a) Generated AWAC Application Architecture (b) Main Modules Actions

4.1 Generated AWAC Application Architecture

The generated Web Application has a three layered architecture as seen in fig. 1a.

- The first layer is the *user interface*, through which 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* and *PRML Evaluator,* see fig 1b). 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 each user will see a different adaptive view of the Website every session s/he browses it. The Website engine captures the events the user performs with his browsing actions and sends them to the *PRML Evaluator* module. This module is responsible of 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 [1]. From the PRML grammar we can generate the syntactic trees which help us to evaluate the rule conditions and perform

them if necessary. Finally to execute the actions of the rules, we have implemented in C# the different actions types that we can find in PRML.

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

4.2 AWAC: PRML Support

PRML was born in the context of the OO-H [11] Web design method to extend it with personalization support (A-OOH). PRML is an ECA rule based language. These rules update the information needed for the adaptation in the User Model and perform adaptation actions over the structure of the Website[4].

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[5]). In Table 1 the personalization actions supported by PRML and the ones implemented in AWAC are shown. Next the actions supported by AWAC are detailed.

Table 1: PRML Support in the AWAC tool

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

[4] Personalization of the presentation is not yet considered.
[5] Note that PRML rules can be attached to nodes or to links of the Navigation Model

- *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 corresponding model of 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. 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 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 shows 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 current 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 current session from the Application database.

The adaptive actions are only performed once during a session not to overwhelm the user, this means the filtered attributes, links, instances and sorted instances will be the same during the present session. However, the desirable option (future work) is that the designer decides when adaptation should take place to fulfil each personalization requirement. Next, by means of a running example, the steps needed for creating and running a Website using AWAC are described.

5 Step 1: Creating the A-OOH Models

To better understand how to generate adaptive Websites with AWAC using A-OOH and 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 2 the Domain and User Model are shown. In the User Model we store different information needed to fulfill the personalization requirements initially specified for the Website:

1. *Users will see fifteen (maximum) recommendations of books in which authors they are most interested in (sorted by interest).*
2. *If the user does not have enough interest in any book author to get personalized recommendations, the recommendations link is not shown:* In order to fulfil the 1st and 2nd requirements we need to acquire and update the user interest in the different authors
3. *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.

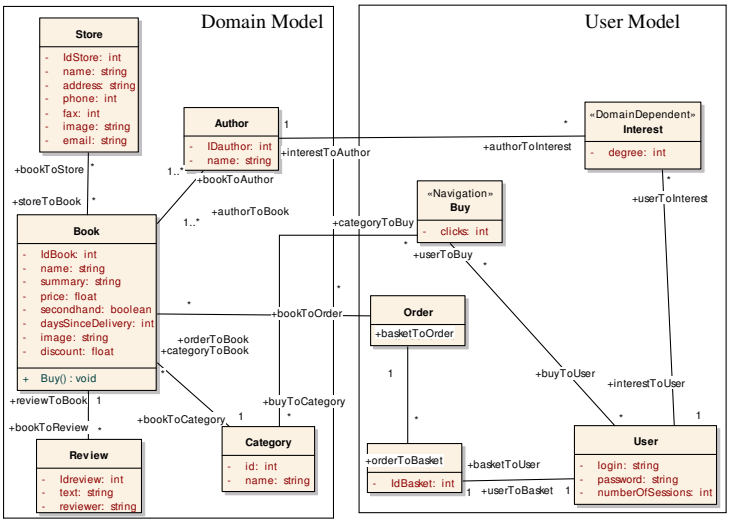

Fig. 2. Domain and User Model for the Online Library

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 3rd 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 3 shows the Navigational Model of the online library. When the user enters the Website (login) 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 3 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 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.

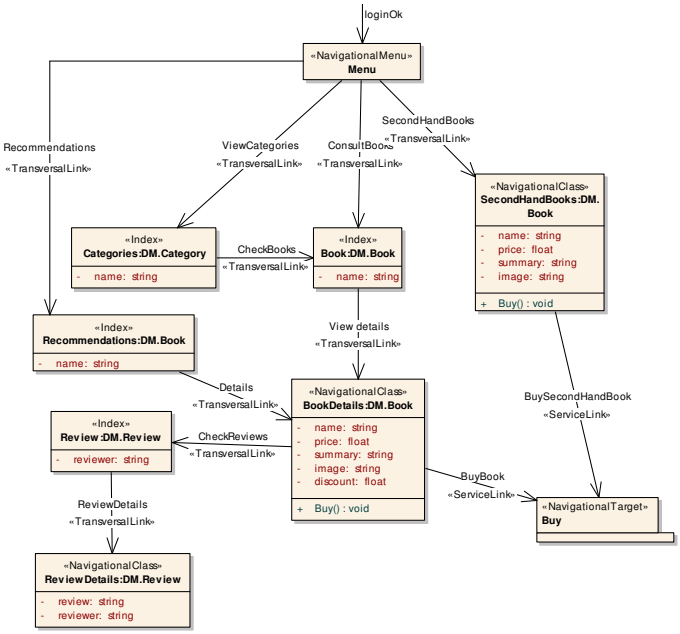

Fig. 3. Excerpt of the Navigational Model

Once the NM is specified the Presentation Model has to be defined. It is captured by one or more Design Presentation Diagrams (i.e. DPDs). There should be one DPD for each NM defined in the system. This diagram enriches the Navigation Diagram described in previously. 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 NM 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.
In Figure 4 the DPD (level 1) for the page that shows the recommendations to the user. This page is defined as a set of layouts and cells. The layouts are also composed of cells. Cells contain interface components that represent the elements of the Web page. 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. Inside this package the Presentation Model attached to the Page Chunk is shown. In this case, the Menu is defined as a page chunk. The final Web page for recommendations generated on basis of these models is in Figure 8.

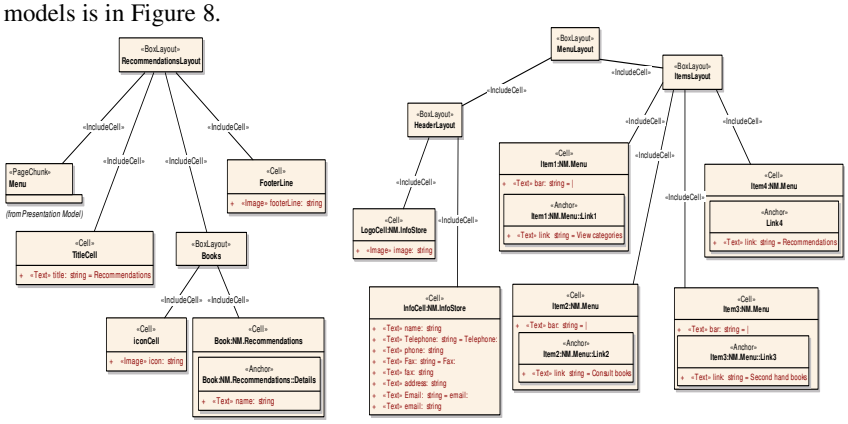

Fig. 4. (a) PM for the Recommendations Page (b) PM for the Menu page chunk

Note that it is not the objective of the paper to explain the A-OOH design models in depth, but to give an overview so the reader can better understand the proposal.

6 Step 2: Adding Personalization using a PRML file

What is left now is defining the Personalization Model in which the adaptive actions to perform over the previously defined set of models are specified. For this purpose we use the PRML language. The basic structure of a rule defined with PRML is the following:

```
When event do
        [Foreach expression]
                [If condition then] 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.
Please note that the purpose of this paper is not to explain the PRML language, for a better understanding of the rules the reader can consult [10].
Now we define the PRML configuration file for our running example[6]. 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 activation of the link *Buy*. The number of books bought along all the sessions is stored in the *buy* class of the User Model by increasing the attribute *clicks* every time the user buys a book.

```
# ACQUISITION SECTION
#RULE:"AcquireInterest"
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
#RULE:"AcquireBuyclicks"
When Navigation.Buy (NM.Book b) do
setContent(UM.User.userToBuy.clicks, UM.User.userToBuy.clicks + 1)
endWhen
```

The **personalization rule section** contains the personalization rules which describe the effect of personalization in the Website. In our example, we have three personalization rules:

- *The first rule* is triggered by the activation of the *Recommendations* link. To fulfil the 1st 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 highest interest (and greater than 100) to be recommended to the user.
- *The second rule* hides the *Recommendations* link if there is no recommendation to show to the user. It is triggered by the *SessionStart* event (i.e. when the user enters the Website).
- *The third rule* is triggered by the activation of the *ViewDetails* link. This rule checks the number of books bought by the user and if this number is greater than 10 it shows the attribute discount of the book to be shown.

[6] Some attributes of the rules have been omitted for simplicity reasons.

```
# PERSONALIZATION SECTION
# RULE:"ShowRecommendations"
When Navigation.Recommendations(NM.Book* books) do
 Sort books orderBy UM.User.userToInterest.degree ASC LIMIT 15 Where
 UM.User.userToInterest.interestToAuthor.ID= books.bookToAuthor.ID
 and UM.User.userToInterest.degree>100
endWhen
# RULE:"HideLink"
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
# RULE:"ShowDiscount"
When Navigation.ViewDetails(NM.Book book) do
      If UM.User.userToBuy.clicks >10 then
       book.Attributes.selectAttribute(discount)
endIf
endWhen
```

7 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 5(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 5(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). It is desirable that the extension of the file is".p" for clarity purposes, but this is not mandatory.
In further versions it will be possible to edit the rules loaded and check if they are syntactically and semantically correct.

4. *Generate 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 7).
As explained in Section 4, 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 and the PRML Evaluator) and the application database.

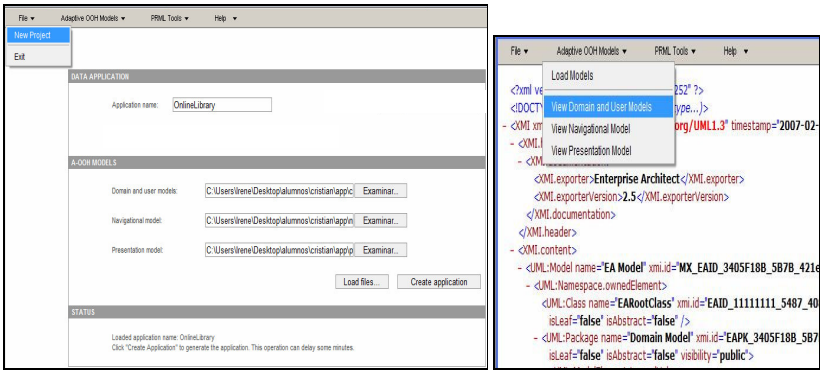

Fig. 5. (a) AWAC tool: loading the A-OOH models (b) View of the models in XMI

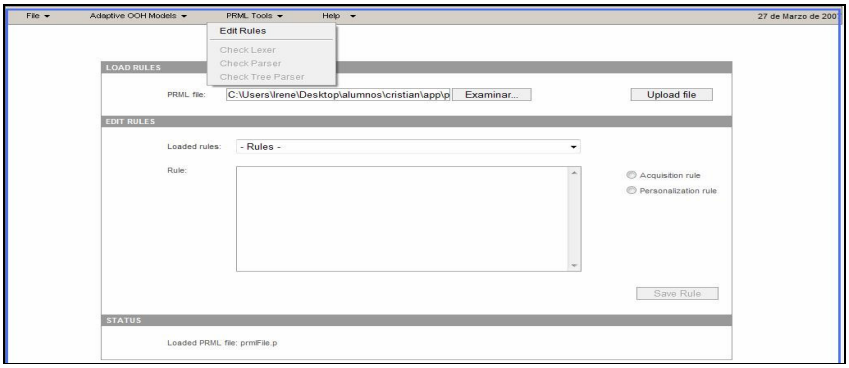

Fig. 6. Loading the PRML file

5. *Deploying and running the Website*

Once generated, the adaptive Web application can be run in a Web server. The adaptive Web pages shown to the users will differ depending on their browsing actions. In Figure 8 the *recommendations* page is shown for two different users. Depending on the user interest (stored in the UM) the books to recommend vary. To properly show the recommendations to the user the AWAC modules generated for managing the personalization (explained in Section 4.1) follow the next steps (see Figure 1b): The *Website Engine* gathers the request of the user for the recommendations page and triggers the user browsing event (i.e. click on recommendations) sending it to the *PRML Evaluator* module. This module checks if there is any rule triggered, and finds the "ShowRecommendations" rule which executes. This rule selects and sorts the corresponding book instances to be shown from the Navigational Model. These recommendations won't change until the next time the user starts a session. This decision has been taken not to overwhelm the user with constant updates.

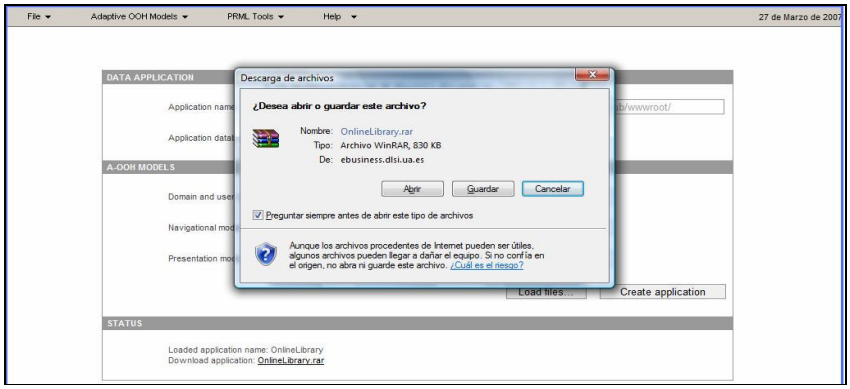

Fig. 7. Generate and download the Web application

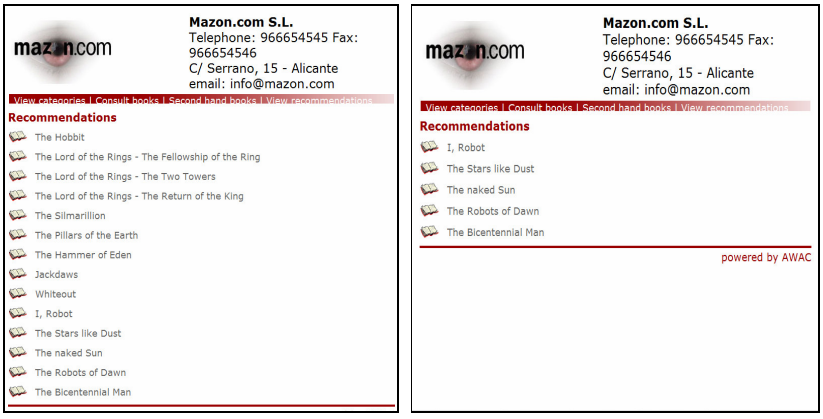

Fig. 8. Running the Website: Recommendations page for two different users

8 Conclusions and Future Work

This paper 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 allow to manage 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.

Some 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. The purpose of this experiment is twofold: to study the satisfaction of the users (in terms of personalization performed, fast response…) and the improvement of the personalization techniques applied. 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 [8] in which we have defined the UML profiles needed for the modelling of the A-OOH diagrams.

References

1. ANTLR, ANother Tool for Language Recognition, http://www.antlr.org/
2. Casteleyn, S.: "Designer Specified Self Re-organizing Websites", Phd thesis, Vrije Universiteit Brussel (2005)
3. Casteleyn, S., De Troyer, O., Brockmans, S.: Design Time Support for Adaptive Behaviour in Web Sites, In Proc. of the 18th ACM Symposium on Applied Computing, Melbourne, USA (2003), pp. 1222 – 1228.
4. Ceri S., Fraternali P., and Bongio A: "Web Modeling Language (WebML): a modeling language for designing Web sites", WWW9 Conf, 2000.
5. Dayal U.: "Active Database Management Systems", In Proc. 3rd Int. Conf on Data and Knowledge Bases, pp 150–169, 1988.
6. Db4o Database For Objects www.db4o.com
7. De Bra, P., Stash, N., De Lange, B., AHA! Adding Adaptive Behavior to Websites. Proceedings of the NLUUG Conference, pp. n-n+10, Ede, The Netherlands, May 2003
8. Enterprise Architect - UML Design Tool, http://www.sparxsystems.com
9. Franciscar, F., Houben G.J., Barna P.:"HPG: The Hera Presentation Generator". *Journal of Web Engineering*, Vol. 5, No. 2, p. 175-200, 2006, Rinton Press
10. Garrigós I., Gómez J., Barna P., Houben G.J.: A Reusable Personalization Model in Web Application Design. International Workshop on Web Information Systems Modeling (WISM 2005) July 2005 Sydney, Australia.
11. Gómez, J., Cachero, C., and Pastor, O.: "Conceptual Modelling of Device-Independent Web Applications", IEEE Multimedia Special Issue on Web Engineering, pp 26–39, 2001.
12. Houben, G.J., Frasincar, F., Barna, P, and Vdovjak, R.: Modeling User Input and Hypermedia Dynamics in Hera International Conference on Web Engineering (ICWE 2004), LNCS 3140, Springer-Verlag, Munich(2004) pp 60-73.
13. Knapp A., Koch N., Moser F., Zhang, G.: ArgoUWE: A CASE Tool for Web Applications. EMSISE03, 14 pages, online publication at http://www.pst.informatik.unimuenchen.de/~kochn
14. Koch, N. and Kraus, A. "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 2002
15. OpenRDF, The SeRQL query language, rev. 1.1, url: http://www.openrdf.org/doc/users/ch06.html.
16. Schwabe, D. and Rossi, G. A Conference Review System with OOHDM. In First Internacional Workshop on Web-Oriented Software Technology, 2001.
17. WebRatio Web Site http://www.webratio.com.
18. XML Metadata Interchange www.omg.org/technology/documents/formal/xmi.htm