Designing Personalized Web Applications

Gustavo Rossi
LIFIA - Facultad de Informática. UNLP,
La Plata
Argentina
gustavo@sol.info.unlp.edu.ar

Daniel Schwabe
Dept. of Informatics, PUC-Rio
Rio de Janeiro
Brazil
+55 21 274 4449
schwabe@inf.puc-rio.br

Robson Guimarães
Dept. of Informatics, PUC-Rio
Rio de Janeiro
Brazil
+55 21 274 44 49
robson@inf.puc-rio.br

Abstract

The goal of this paper is to argue the need to approach the personalization issues in Web applications from the very beginning in the application's development cycle. Since personalization is a critical aspect in many popular domains such as e-commerce, it important enough that it should be dealt with through a design view, rather than only an implementation view (which discusses mechanisms, rather than design options). We present different scenarios of personalization covering most existing applications. Since our design approach is based on the Object-Oriented Hypermedia Design Method, we briefly introduce it, emphasizing the way in which we build Web application models as object-oriented views of conceptual models. We show how we specify personalized Web applications by refining views according to users' profiles or preferences; we show that an object-oriented approach allows maximizing reuse in these specifications. We discuss some implementation aspects and compare our work with related approaches, and present some concluding remarks.

1. Introduction

Building personalized Web applications, i.e. those applications that are responsive to the individual needs of each user, is a challenging task. It involves a myriad of different technologies that range from simple database views to software agents and collaborative filtering algorithms. Personalization has become hype in areas such as electronic commerce, and we can find hundreds of applications that claim to be fully customizable to different user profiles or individuals. The number of possible personalization variants seems countless. As with other Web features, a great variety of technologies and systems have been developed and are available in the market [2], but little or no attention has been paid to the process of modeling and designing personalized Web applications (an interesting exception can be found in [3]).

In this paper, we claim that adopting a design-centered view of personalization allows us to better understand the fundamental mechanisms we are using. Consequently, we cannot only build the specific (personalized) aspects of these applications, but mainly reuse those design aspects that are common to most users.

We show that focusing on which design abstractions are necessary to build highly customizable applications allows modeling most personalization features with a few simple design constructs. In addition, a clear understanding of basic modeling and design mechanisms for personalization can help us reason over the development process and uncover reusable patterns, components or sub-systems for the key personalization styles or algorithms.

The discussion is cast using the Object-Oriented Hypermedia Design Method (OOHDM) [16] approach to personalization, discussing the design primitives we added to our basic object-oriented notation in order to specify group or individual customization in an abstract way. Although we center our discussion on OOHDM mechanisms, designers building personalized Web applications employing other approaches can easily use the ideas in this paper.

The structure of the remainder of the paper is as follows: we first analyze different scenarios of personalization that can be found today in the Web. The purpose of this section is to use those scenarios as examples during the whole paper. We next introduce the basic ideas of OOHDM, in particular the separation among conceptual and navigation modeling and the design constructs used in OOHDM to build customized applications. We briefly show how we can map personalized designs into running applications. Finally, we compare our approach with others and discuss some further issues such as personalization patterns and frameworks.

2. Scenarios of Personalization

Although it seems impossible to classify all the existing approaches to personalization, using a simple conceptual framework allows us to show the main differences between most of them. We consider that Web applications are hypermedia applications [16] in the sense that users navigate a hypermedia information space composed of nodes connected by links. The main difference between a "traditional" static hypermedia application and most Web applications is that the latter may involve some business logic (application functionality). In addition, users may alter information while navigating - adding products to a cart for example. There are thus two approaches to characterize personalization: analyzing how the underlying application logic may change for each user or analyzing what may change in the information space the user perceives. We will use the second approach, i.e. we will focus on the structure and contents of the nodes and link topology. It is obvious however, that both aspects are strongly related, as we will show later. Additionally, we will also address in Section 4 how an application's logic flexibility - for example assigning different recommendation algorithms to different users or using intermediaries - can be easily built into our framework.

In this context, we can basically personalize the linking topology or the contents of individual nodes. For the sake of simplicity, we discuss each of them in a separate sub-section. It should be clear to the reader that both kinds of variability can be, and usually are, combined.

WWW10, May 1-5, 2001, Hong Kong.
ACM 1-58113-348-0/01/0005.

2.1 Link Personalization

This strategy involves selecting the links that are more relevant to the user, changing the original navigation space by reducing or improving the relationships between nodes. E-commerce applications use link personalization to recommend items based on the clients buying history or some categorization of clients based on ratings and opinions. Users who give similar ratings to similar objects are presumed to have similar tastes, so when a user seeks recommendations about products, the site suggests those that are most popular for his class, or those that best correlate with the given product for that class. Link personalization is widely used in www.amazon.com (See Figure 1) to link the home page with recommendations, new releases, shopping groups, etc. that are personalized. Amazon.com has taken this approach to an extreme by building a "New for you" home page and presenting it to each user, with those new products in which he may be interested.

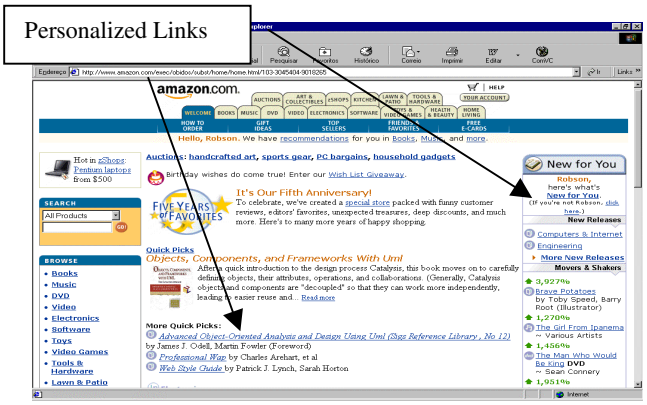


Figure 1: Using Link personalization in www.amazon.com

The same kind of personalization is often found in paper review applications. Each reviewer is presented with a set of links to the articles he will review, which may have been assigned manually by the PC chair, or just computed using a description of the reviewer's expertise. Implementing this latter example of personalization is easier than the previous one, even though their specifications are similar.

2.2 Content Personalization

We can say that content is personalized when nodes (pages) present different information to different users. The difference with link customization is subtle since when links are personalized, part of the contents (the link anchors) present different information. However, we will refer to content personalization when substantive information in a node is personalized, other than link anchors. Content personalization can be further classified into two types: node structure customization and node content customization.

Structure personalization usually appears in those sites that filter the information that is relevant for the user, showing only sections and details in which the user may be interested. The user may explicitly indicate his preferences, or it may be inferred (semi-) automatically from his profile or from his navigation activity. For example, in my.yahoo.com or in www.mycnn.com users choose a set of "modules" (from a large set including weather, news, music, etc…) and further personalize those modules choosing a set of attributes of the module to be perceived. Some "automatic" customization may occur by using the zip code of the user, for instance to select which sport events he may be interested in. The approach followed in these applications is that the user should be able to "build" his own page; even layout may be customized. In Figure 2, we show an example of structure customization in my.yahoo.com.

Figure 2: Structure customization in my.yahoo.com

The objectivity in WAP portals can be improved with the same approach. In the Infospace application (www.infospace.com), the user can customize the content and the content provider, making a kind of content syndication. In this way each customer navigates only through the information he desires, improving the usability of the site. In Figure 3, we show the customization and its result for the WAP phone.

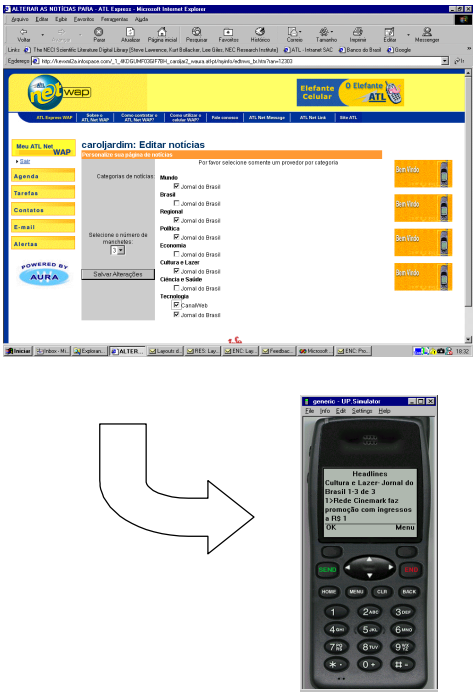
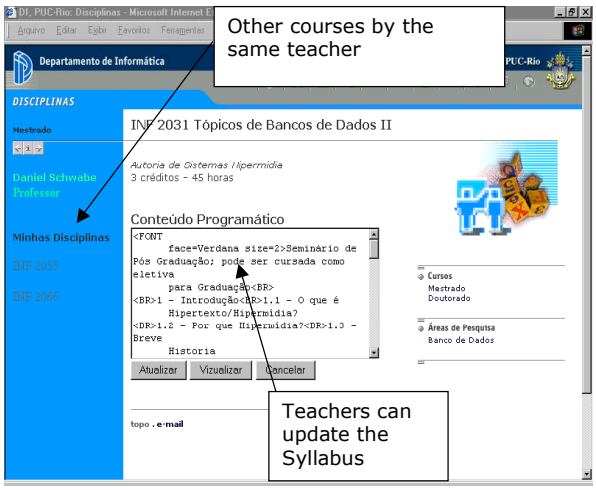


Figure 3: Personalization in WAP Portals

Applications in which different user roles have different access rights or authorizations provide another good example of structure customization. For example suppose an academic application where teachers and students have different tasks to perform; teachers need to access their class schedule to update its contents and students have to access the classes that are available for enrollment, depending on their GPA.



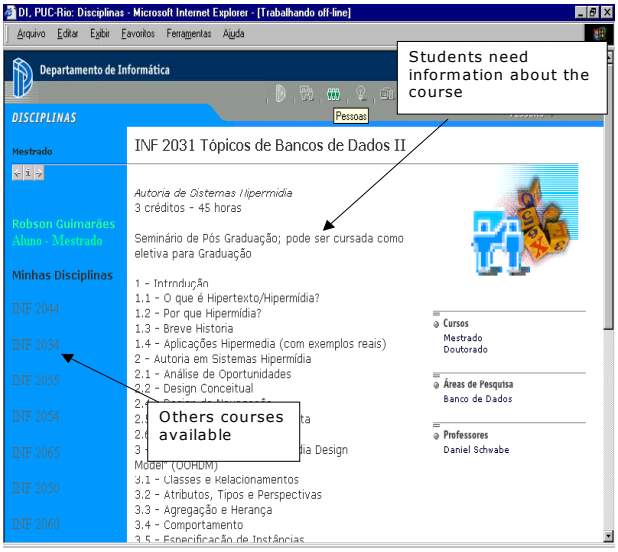


Figure 4: Structure Customization according to users' roles

When a teacher accesses a class node, he can update the class information (e.g. the syllabus), so it is important to make the update button available for the classes for which he is responsible. On the other hand, the student needs to access the syllabus, the course location, the course program, and of course, he cannot modify the site. See Figure 4.

Another difference is the links to related information, in each case. For the teacher it is relevant to provide links to "other courses he teaches", whereas for the student it is relevant to provide links to other "courses that he may take".

Node content personalization occurs when different users perceive different values for the same node attribute; this kind of content personalization is finer grained than structure personalization. A good example can be found in online stores that give customers special discounts according to their buying history (in this case the attribute price of item is personalized). There are many good examples of node content personalization in intranet applications, where employees' role and needs determine the tailoring of information they see.

For example in the ATL (a mobile phone company in Rio de Janeiro) intranet, different sales channels receive different, customized information about business procedures. When a call center attendant looks up information about phone repairs he will receive the address of a repair center; when a repair center employee looks up information for the same procedure, he will receive repair instructions for the phone, as shown in Figure 5. This type of personalization must be designed from the beginning, capturing the personalization rules for the different user groups that are identified.

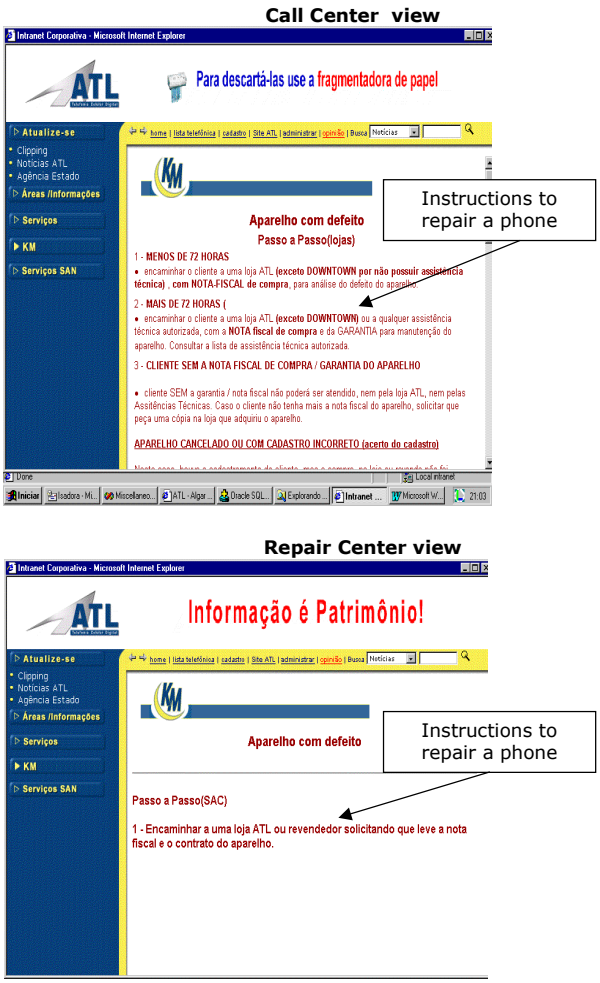


Figure 5: Node content personalization in the ATL intranet

3. The OOHDM Design Approach

OOHDM is a model-based approach for developing Web applications. We have used it for years to build different kinds of applications ranging from websites to complex e-commerce software.

The key concept in OOHDM is that Web application models involve a Conceptual, a Navigational Model and an Interface Model [16]. Those models are built using object-oriented primitives with syntax close to UML [20].

The concern of the conceptual model is to represent domain objects, relationships and the intended applications' functionality. In an electronic store for example, the conceptual model will contain core classes such as Product, Order, Customer, etc. with their corresponding behaviors. Notice that in some Web applications such as e-commerce, these behaviors may be extremely complex and specifying them with an object-oriented notation is valuable even though the implementation may involve different, possibly non object-oriented tools. Most personalization mechanisms involve dealing with objects and algorithms that are expressed as part of the conceptual model. In Figure 6, we show a simplified conceptual model of an electronic store. Objects of the class Customer will be responsible to process requests related with individual customizations as we show in Section 4. The conceptual model in OOHDM subsumes the class model in traditional object-oriented methods. Being based on UML, it can be obviously complemented with other UML models using use cases, sequence diagrams, etc.

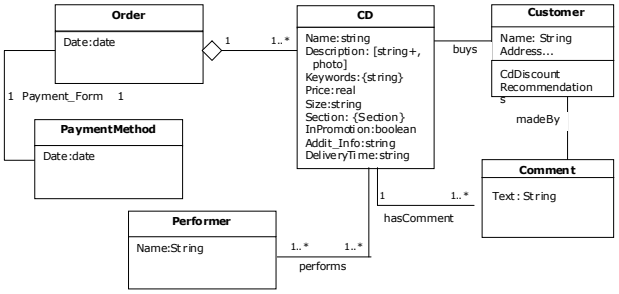
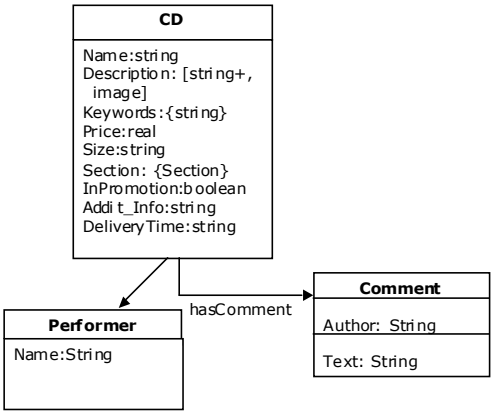


Figure 6: Conceptual Model of CD store

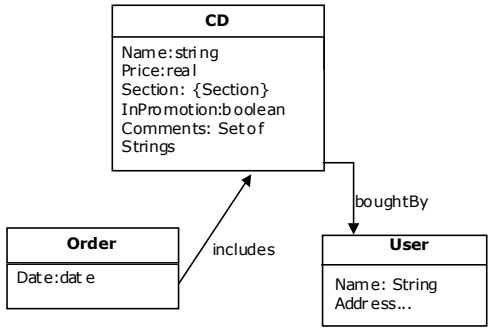
In the OOHDM approach, the user does not navigate through conceptual objects, but through navigation objects (nodes), as we consider Web applications as being hypermedia applications built on top of the conceptual model.

Nodes are defined as views on conceptual objects, using a language that is similar to object database view-definition approaches [10]. Nodes can be atomic (i.e. their attributes are primitive types or anchors) or composite. Composite nodes express aggregations of simpler nodes, such as usually found in home pages or portals. Nodes contain attributes (that are usually perceived in the interface) and anchors for links (that may or not be perceived in the interface).

Nodes are complemented with links that are themselves specified as views on conceptual relationships. The navigational schema shows the node and link classes that comprise the navigational structure of the application. Notice that in some way, the navigational model allows to specify different Web applications for the same conceptual model. In particular, regarding personalization, OOHDM provides one easy way to build customized Web applications by reusing the conceptual model. We do this by building different navigational models (views) for each user profile. In Figure 7, we present parts of two different navigational schemas for the previously shown conceptual schema.



7.a. Customer Profile View



7.b. Manager Profile View

Figure 7: Different Navigation Schemata for user and manager profiles

Notice that each one of above schema presents the user with different features and links (inducing different navigation topologies) according to the needs of the specific user profile. For example, while customers may navigate through comments, managers only see comments as attributes of CDs; in the same way, customers may access information about performers, but only managers may navigate the information about users. In Figure 8, we show part of the specification of node CD for both profiles. In the Customer profile the *comments* attributes is an anchor to Comment nodes (via the *Comments* link type). This means that the user will have to navigate to those nodes to read comments. Meanwhile, in the Manager profile, the *comments* attribute is just a list of strings; we could have even omitted this attribute, as perhaps managers are not interested in reading comments.

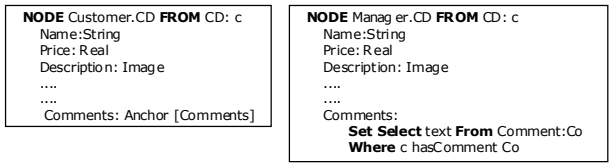


Figure 8: Specification of Node CD for Customer and Manager profiles

The complete syntax for Node and Link definition can be found in [16]. Two problems arise from the previously shown specifications: First, how can we improve reuse in the

navigational schema, e.g.: how do we express the common aspects of both profiles (customer and manager) only once, and which mechanism do we use for expressing variability? Second, how can we express individual personalization, i.e. the fact that two different users may pay a different price for a CD?

In OOHDM, a Context Schema shows the navigational contexts and access structures (indexes) in the application complements the Navigational Schema. A navigational context is a set of objects that may be explored in some order, e.g., sequentially. For example: books by an author, CDs by a rock band, etc. Access Structures act as indexes to group of related objects, specified by indicating the target objects and the selector to be used in the index. Indexes are key customizable artifacts because, as we showed in Section 2, many personalization scenarios involve defining customized indexes.

Finally, we specify the abstract interface model that indicates the look and feel of navigation objects. Separating the interface from the navigation specification allows us to cope with varying interface technologies in a modular way. For example, given a particular navigation model we can specify different interfaces – for a browser or for a variety of mobile devices such as phones, palm tops, etc. We also use an object-oriented formalism, called Abstract Data Views ADVs [5] that act as Observers [Gamma95] of nodes. For the sake of conciseness, we do not address interface personalization in this paper, though it is handled with the same approach as presented in the following section.

Summarizing, OOHDM provides some hot-spots in which we can specify customized structures and behaviors (the idea of hot-spots is borrowed from the domain of application frameworks [7]):

- in the conceptual model: by explicitly representing users, roles and groups and by defining algorithms that implement different (business) rules for different users.
- in the navigational model: by defining completely different applications for each profile, by customizing node contents and structure and by personalizing links and indexes.
- in the interface model: by defining different layouts according to user preferences or selected devices.

In section 4, we show how we can specify different kinds of personalized behaviors by relating each of the above hot spots with the examples presented in Section 2.

4. Designing Personalized Views

One of the key issues of the OOHDM meta-model is that we emphasize expressiveness and openness using a reduced set of primitives. To do this, we adhere to the best object-oriented design practices, in particular an appropriate use of Design Patterns [Gamma95]. We strongly claim that we can build high quality personalized software only if we design that software with flexibility and extension in mind from the beginning. Moreover, many personalization strategies are simply by-products or design refinements of more abstract ones. In most cases, we need to have a model of the user or group of users. User or profile modeling is a challenging task and it may require the systematic application of complex techniques (See [19] for a broad range of approaches). In this paper, we are not concerned with which algorithms are used to assign a profile to a particular user, but mainly with the way to express those algorithms as parts of a complete Web application model. We next present each customization-oriented feature of OOHDM with an example.

4.1 Static Customization

OOHDM supports the design of different applications (navigational views) for the same conceptual model, and of different user interfaces for the same application. In this way, we can customize the conceptual model to different user profiles or roles.

This kind of customization, as shown in Figure 7, results from a direct application of the pattern Observer to decouple design concerns. More specifically, we decouple three aspects: the base information and behavior (conceptual) from what the user perceives (navigation) and how he perceives it (interface). This kind of customization is static, in the sense that it is fully defined at design time. Determining which user is accessing the application requires that the user either identifies himself explicitly at the beginning, or that different entry points are given (externally to the application) to different user classes. We next address different kinds of dynamic customization.

4.2 Link and Content Personalization

Personalizing content and links in a Web application is by far the most popular way of individual customization currently found in the Web, with many different variants. For instance, recommendations in Amazon.com are based on the history of the user; links to specific Purchase groups are built from user personal data.

In Figure 9, we introduce the variable *user* in the specification of Figure 8 to assign a personalized price for a CD according to the discount policy of the company; the expression between brackets indicates how the value of the variable is computed. Notice that *user* refers to an instance of class Customer in the conceptual model (heading of Node definition). It provides the intended behavior (CDdiscount, see Figure 6). The method CDdiscount calculates the discount rate according to some data related with the user's buying history, his category, etc. The variable *subject* meanwhile refers to the corresponding CD conceptual object.

```
NODE Customer.CD FROM CD:c, user:Customer
    Name:String
    Price: Real [subject.price – user CDdiscount]
    Description: Image
    ....
    ....
    Comments: Anchor [Comments]
```

Figure 9: Personalizing content in a node

The expression *subject.price – user CDdiscount* will be executed during node initialization. Notice that in this case the object *user* will be sent as a parameter to the corresponding method (*CDdiscount*).

In Figure 10, we show how we get individual recommendations by mapping the definition of the Recommendation link in the home page to the CDs that are recommended for that user. The expression *user recommendations* assumes that Class Customer implements an interface (in the sense of Java programs) that answers the message *recommendations* (See Figure 6).

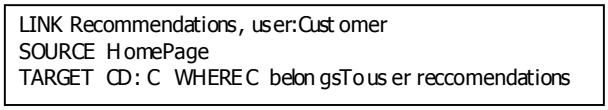


Figure 10: Link personalization in OOHDM

The openness of the OOHDM approach resides in that it makes easy to apply well-known techniques for building generic designs. A first example is the representation of more than one user role in the conceptual model (Figure 6). Class *Customer* thus may be a sub-class of a more abstract class *Role* and this allows us to define node features that are valid for all instances of *Role* or one of its sub-classes.

A more interesting example arises if we want to improve the use of recommendation algorithms. We can model the assignment of different algorithms to different users by using Strategies [8] as shown in Figure 11. In this way, we can get a further level of decoupling: recommendation algorithms are decoupled from the user object. Using Strategies to model this kind of personalization allows assigning a different recommendation algorithm (an instance of a RecommAlgorithm sub-class in Figure 11) to each user. The corresponding user instance just delegates the message to the current algorithm object. In Figure 12, we present the UML sequence diagram explaining how this decoupling works.

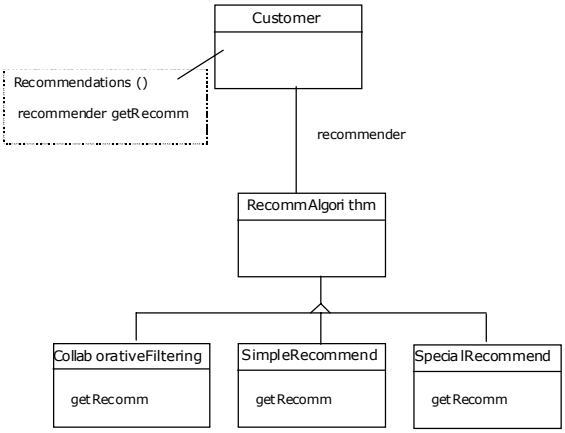


Figure 11 Decoupling users from Recommendation algorithms

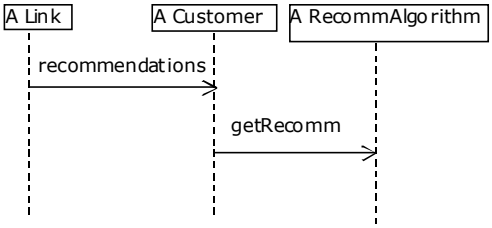


Figure 12: Sequence Diagram for recommendation strategies

In the same way the use of third party products (such as Netperception), Internet Services [15] is easily modeled by using Adapters as shown in Figure 13.

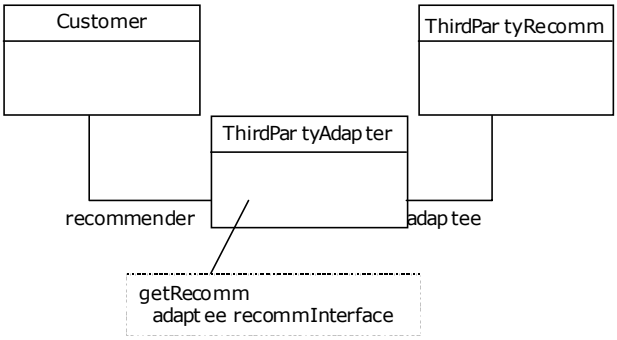


Figure 13: Accommodating third party products

In Figure 13, the Adapter object (ThirdPartyAdapter) acts as a Wrapper on the "external" product and "adapts" its interface to the existing protocol. Notice that in this way the Customer class can be kept unchanged regardless of the specific features or implementation style of the third party product. In Figure 14, we show the corresponding sequence diagram.

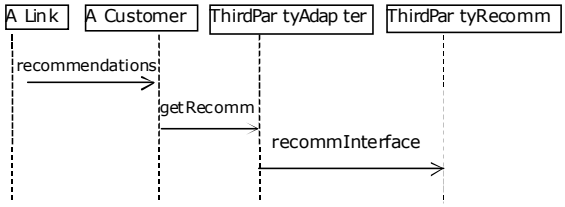


Figure 14: Sequence diagram for adapting third party products

Notice that in the actual implementation, the binding of the variable *user* with the proper instance is done when the user is identified and authenticated. In an object-oriented implementation, this process involves only a couple of trivial messages between objects. In the OOHDM framework we allow the designer to define other variables such as *profile, group*, etc which may be used in the same way as *user* by declaring them in the Node definition heading.

4.3 Structure Customization

As discussed in Section 2, many Web sites allow the user to select which contents he wants to see from a repertoire of options (most of them also allow customizing the interface lay-out as well). For example, in my.yahoo.com there are two levels of personalization: first, which modules the user will get in his site (e.g., Weather, Headlines, Financial, etc) and then which information he wants to see within each module (cities, types of news, particular stock quotations, etc). In Figures 15 and 16, we show (part of) the OOHDM specification for this kind of structural personalization.

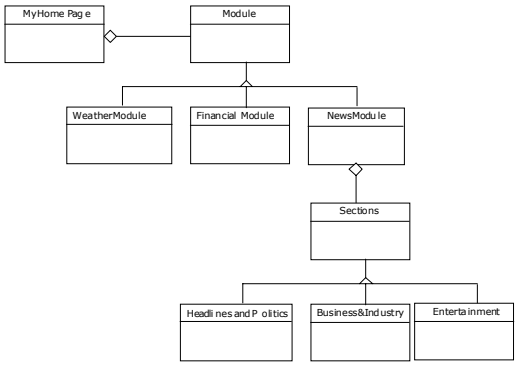


Figure 15: Basic Navigational Schema in my.yahoo.com

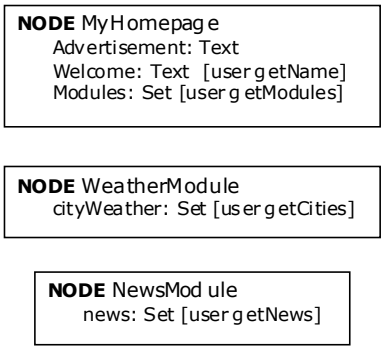


Figure 16: Customizing node's structure

The navigational class schema in Figure 15 shows MyHomePage as a composition of modules (that are themselves instances of a sub-class of Module), containing different kinds of information. The corresponding user object in the conceptual model provides an interface for retrieving his user profile, in this case the modules he chose, as well as the structure of these modules (Figure 16). Each Module sub-class has a different structure – cities and their weather in the Weather Module, list of links to news in the NewsModule, etc… – so we cannot factor out attribute definitions and include them in the Module Node. Notice that the News Module is itself an aggregation of other sub-modules. Using this style of specification, we can go even further and personalize more specific features, such as the units in which temperatures are given (see mycnn.com).

The conceptual model will contain a class User and the corresponding information objects for Modules. As discussed in [12] the structure of the user preferences database may be itself rather complex and may need further decomposition in the conceptual model.

In the case of applications acting as intermediaries (such as Yahoo News with respect to news provided by Reuter), we only need to add another level of observation, i.e. objects in the application represent views on the information provider (Figure 17). In the corresponding OOHDM conceptual model, we define one conceptual class for each intermediary. The relationship *observes* abstracts different kinds of intermediary functions, such as customization, filtering, annotation, etc [11].

In the case of the intermediaries, we may assume "push" or "pull" models, since the communication between them may be implemented using a proprietary protocol, as customary with news providers. The sequence diagram for this Class structure is similar to the one in Figure 14; the main difference is the intent of the relationship among intermediaries and providers with respect to the adapter's example.

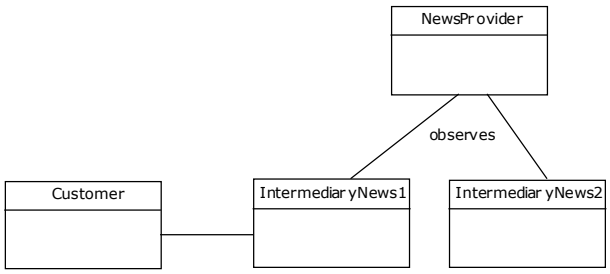


Figure 17: Modeling intermediaries as Observers

4.4 Context Personalization

Personalizing navigational contexts is critical when the same information (node) can be reached in different situations. As mentioned before, a navigational context is a set of nodes that usually share some property. For example in a Conference Paper Review Application, it is possible to access papers in different contexts: the whole set of papers, papers that were reviewed by a person, papers in a particular topic, accepted papers, etc. Notice that one paper may appear in different sets and that different users may have different access restrictions according to their role in the Review application. When the program chair accesses a paper in this system, he can navigate and see the reviews/reviewers of this paper, but when a reviewer accesses the same information, he will not able to follow that link (that simply will not exist for him).

In OOHDM, we use a simple declarative specification to indicate

1-which nodes are contained in a context and

2-which user or user profiles are allowed to view a node in a particular context.

The set of nodes are specified using expressions similar to the one in Figure 10, while the access restrictions (if any) are specified in Context Cards, indicating either a user profile (such as PCChair) or an expression indicating a condition on the *user* object (See [16]). Figure 18 shows a portion of an OOHDM context diagram for this example, and Figure 19 shows the context cards related to Paper.

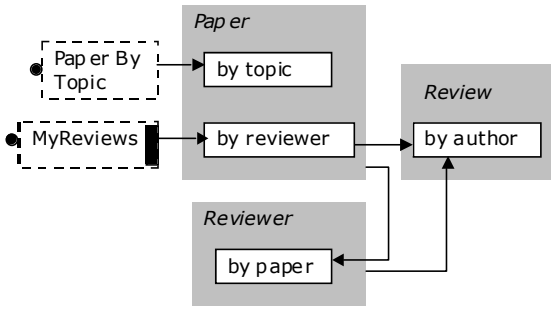


Figure 18: Navigation Diagram of Paper Review system scenario

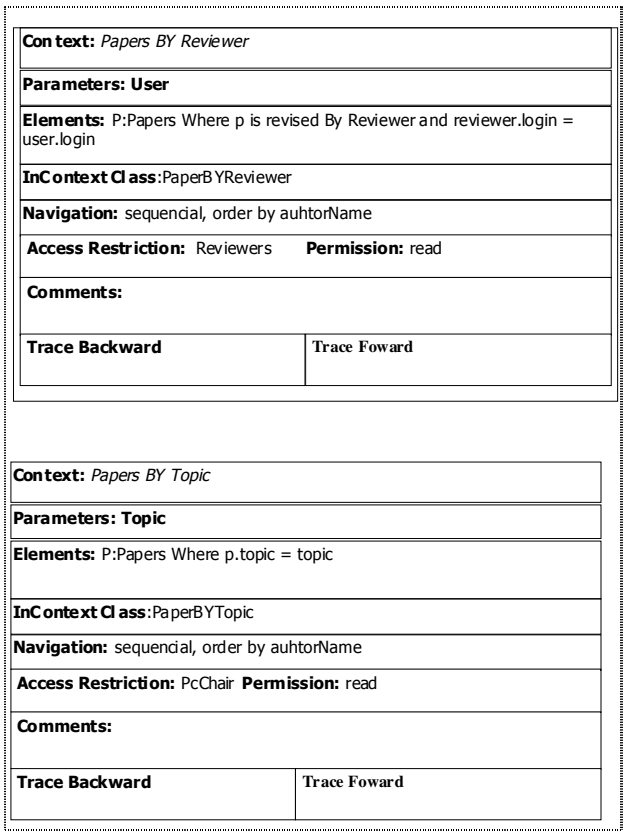


Figure 19: Context specification card

4.5 Reusing Specifications

Different user profiles may share part of the same specification and, in fact, share the same information in the final application, in many cases. For example, in the online store example (see Figure 8), both customer and manager specifications of CD Node exhibit similar information with just small variations.

In Figure 20, we show the specification of the base information and the corresponding extensions for user profile customer and manager. The precise semantics of the extension classes as they appear in the OOHDM specification is that they act as Decorators [8] of the extended class.

NODE CD **FROM** CD: C Name: String Price: Real	
NODE Customer.CD **EXTENDS** CD Description: Image Comments: Anchor [Comments]	**NODE** Manager.CD **EXTENDS** CD Comments: **Set Select** text **From** Comment:Co **Where** C hasComment Co

Figure 20: Extending a Node specification for different user profiles

This same approach can be used for adding specialized behaviors for each user profile. For example while Customers can add a product to the shopping basket, Managers can eventually change the price of a product.

Notice the subtle difference between defining the Customer and Manager CD nodes as decorations on CD with respect to defining them as sub-classes. The OOHDM solution is more flexible than using inheritance, since one may sub-classify CDs independently of Manager and Customer views. A deeper discussion on the differences between decorating and sub-classing can be found in [8].

In the same way, we can use this approach for specifying different access rights for different user roles. For example, in the Conference Paper Review application, we can define a Node class Paper with some basic information and specify that papers be linked to reviews/reviewers only in the PcChair.Paper extension, by defining the appropriate anchor and other related attributes. In Figure 21, we show this example of role customization.

NODE Paper **From** Paper:p Name:String Authors: String Abstract: Text	**NODE** PCChair.Paper **EXTENDS** Paper reviews: Anchor [ReviewedBy] additionalComments: Text

Figure 21: Expressing different access rights with extensions

5. From Design to Implementation

The standard way to obtain dynamic behavior is by using HTML pages with calls to server side procedures written in a variety of scripting languages, or to components in some framework. This type of solution leads to high development costs, as it is necessary to develop each solution from the beginning. In this context, a personalization solution will lead to calls that pass the information about the user and the navigation models to procedures, to compute the end result.

In a higher abstraction level, we can mention the so-called site management systems such as MS SiteServer, Broadvision OnetoOne and Vignette StoryServer. These systems offer an interface between server-based repositories to controlled client web delivery, separating content and business rules. The content stays in a relational database, and the business rules are stored into proprietary components that wrap the association between the users, roles and the appropriate content. Our approach is adequate to this implementation strategy because the main assumption is to design what to personalize and the corresponding conceptual rules based on the conceptual model. The main difficulties in this approach are due to eventual limitations in the expressive power of the proprietary components.

Although feasible, the use of such systems is still limited to larger websites, due to cost and complexity. In addition, none of these systems directly supports the notion of contexts as defined in OOHDM. We have been using the OOHDM Web environment [17] to support the implementation of OOHDM designs. This environment is very similar to the systems mentioned previously, but with components that directly understand OOHDM primitives. Special data structures represent a design; specific library of functions access and manipulate these structures, allowing automatic generation of index structures, object manipulation and context navigation. Currently, it only allows a very simple form of role based access control for nodes within contexts.

As discussed in sections 3 and 4, the various mechanisms in OOHDM directly support many types of personalization, and therefore are available through the OOHDM Web environment. A more recent development is the upgrading of this tool to support a complete specification of user customization rules, and the corresponding function libraries to implement them.

A different architectural approach is the use of proxies that host programs that implement filters providing the personalization functions. These proxies serve as intermediaries between the user and the Web, customizing the user's view of the web according to various rules, policies and preferences [1].

6. Related Work

Personalization is an important topic for the Web industry. However, design issues related with personalization are just being introduced in the Web community. An interesting approach has been discussed in [6]. The Web Site Design Method (WSDM) focuses on the construction of "Audience-driven" Web applications. Requirements for each potential user profile are systematically gathered and a class diagram of user profiles is built. Different navigation tracks are specified for each audience though there is no notation for expressing individual differences.

More recently, OOHDM has been extended with a requirements gathering phase that uses a new graphical notation and model, called User Interaction Diagrams [21], [9]. Requirements are gathered collecting scenarios that are subsequently abstracted into Use Cases. The analysis of these Use Cases determines the user profiles and tasks that the application will support. Although we do not yet explicitly extract personalization information, this analysis can be extended straightforwardly to achieve this.

Another interesting approach for dealing with personalization is presented in [3],[4]. Using the WebML modeling language and its supporting tool (Torii) it is possible to specify the structure of the application's information base (the Structure Schema), the structure of nodes (Composition Model) and the navigation model. WebML allows representing well known Web patterns and supports data derivation and user modeling. Personalization in WebML is expressed using event-condition-action rules.

The main difference with the OOHDM approach is that WebML claims to be data-oriented while OOHDM is object-oriented. This difference may seem trivial for simple applications, i.e. applications in which the behavior can be modeled by using just database updates and triggers. WebML rules are concise and elegant and can be easily mapped to a database implementation. However, we think that applications in which personalization patterns may involve either sophisticated behaviors or interaction with other systems, such as Internet services or intermediaries, require a more expressive specification language. Nevertheless, as the approaches are similar, WebML can be used as part of the implementation step in the OOHDM life cycle, for example when mapping the design to a relational database.

Finally, it is worth mentioning the relationship of our kind of personalization specification with current approaches for providing services on the Internet such as Application Services Providers (such as www.aspstreet.com). It is reasonable to envision that personalization algorithms will be provided as services (such as www.moviecritic.com) and applications will just use these services as "external" components. In these cases, a design notation is necessary; it should be able to express those design decisions that make "internal" and "external" components interact seamlessly. The OOHDM approach for personalization acts as an open framework not only for specifying "ad-hoc" customization but also eventually for documenting the use of different personalization products.

7. Conclusions

In this paper, we presented the OOHDM approach for specifying and designing personalized behaviors in Web applications. The main goal of this paper is to motivate a design-oriented discussion of personalization. We believe that by clearly understanding the design structures involved in the process of building a personalized application we can obtain more flexible and evolvable systems.

We have shown how we use our notation in a broad range of customization cases. For example, we can build different Web applications for different profiles by just reusing a conceptual schema. A finer grained personalization can be obtained by specifying individual contents such as recommendations, customized prices, etc. We have shown that our notation and the underlying design framework allow us to obtain concise specifications by reusing existing ones. 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.

We are currently extending our Web design pattern language [13] with personalization patterns, i.e. patterns that capture recurrent customization structures and their underlying designs. We are also adapting our Web framework notation [18] to include personalization variability as hot spots in the framework structure. Building generic personalization engines is an interesting trend as discussed in [15], and we aim at representing them not only as external services but also in such a way that a designer can reason on their structures to extend them to different domains and roles. Analyzing personalization from a software engineering point of view will allow our community to design more modular and easy to maintain and extend customized applications.

8. References

[1] Barrett, R., Maglio, P. P.: "Intermediaries: New places for producing and manipulating web content". In Proceedings of the Seventh International World Wide Web Conference (WWW7), Brisbane, Australia (1998).

[2] Communications of the ACM. Special Issue on Personalization, August 2000, Volume 43, Number 8.

[3] S. Ceri, P. Fraternali, S. Paraboschi: "Data-Driven One-to-One Web Site Generation for Data-Intensive Applications". Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999, pp 615-626.

[4] S. Ceri, P. Fraternali, S. Paraboschi: "Web Modeling Language", (WebML): a modeling language for designing Web sites. Proceedings of the 9th. International World Wide Web Conference, Elsevier, 2000, pp 137-157.

[5] D. D. Cowan, C. J. P. Lucena: "Abstract Data Views, An Interface Specification Concept to Enhance Design for Reuse", IEEE Transactions on Software Engineering, Vol. 21, No. 3, March 1995.

[6] O. De Troyer, C. Leune: "WSDM: A User-Centered Design Method for Web Sites", Proceedings of the 7th International World Wide Web Conference, Elsevier, 1998, pp 85-94.

[7] M. Fayad, D. Schmidt and R. Johnson (editors): "Building Application Frameworks", Wiley, 1999.

[8] E. Gamma, R. Helm, R. Johnson and J. Vlissides: "Design Patterns: Elements of reusable object-oriented software", Addison Wesley, 1995.

[9] Guell, N., Schwabe D., Vilain, P.: "Modeling Interactions and Navigation in Web Applications", Lecture Notes in Computer Science 1921, Proceedings of the World Wild Web and Conceptual Modeling '00 Workshop, ER'99 Conference, Springer, Salt Lake City, 2000. ISBN 3-540-41073-2.

[10] W. Kim: "Advanced Database systems", ACM Press, 1994.

[11] P. Maglio, R. Barret: "Intermediaries Personalize Information Streams". In [CACM'00], pp 96-101.

[12] U. Manber, A. Pattel, J. Robison: "Experience with Personalization on Yahoo!". In [CACM2000], pp 35-39.

[13] G. Rossi, D. Schwabe, F. Lyardet: "Improving Web Information Systems with Navigational Patterns",

[14] Proceedings of the 8th International Conference on the World Wide Web, Elsevier 1999, pp 589-600.

[15] S. Sarawagi, S. Nagaralu: "Data Mining Models as Services on the Internet". SIGKDD Explorations, ACM Press, June 2000, pp 24-28.

[16] D. Schwabe, G. Rossi: "An object-oriented approach to web-based application design". Theory and Practice of Object Systems (TAPOS), Special Issue on the Internet, v. 4#4, pp. 207-225, October, 1998.

[17] Schwabe, D., Pontes, R. A., Moura, I.: "OOHDM-Web: An Environment for Implementation of Hypermedia Applications in the WWW", SigWEB Newsletter, Vol. 8, #2, Junho de 1999.

[18] D. Schwabe, G. Rossi, L. Esmeraldo, F. Lyardet: "Engineering Web Applications for reuse". To appear, IEEE Multimedia, Spring 2001.

[19] User Modeling Home page: www.um.org

[20] Rational. UML Documentation www.rational.com

[21] Vilain, P., Schwabe, D., Souza, C.S.: "A Diagrammatic Tool for Representing User Interaction in UML", Lecture Notes in Computer Science, Proc. UML'2000, York, 2000.