Extending WebML for Modeling Multi-Channel Context-Aware Web Applications

Stefano Ceri, Florian Daniel, Maristella Matera
Dipartimento di Elettronica - Politecnico di Milano
P.zza Leonardo da Vinci, 32 - 20133 - Milano, Italy
{ceri,matera,florian}@elet.polimi.it

Abstract

This paper focuses on the conceptual modeling of multi-channel, context-aware Web applications, and in particular proposes some solutions conceived within the WebML method. WebML is a conceptual model for data-intensive Web applications, which already offers some constructs for one-to-one personalization and multi-channel delivery. In this paper we introduce some new extensions that will allow representing a context model at data level, and exploiting it at the hypertext level for offering customized services and contents, accessible through multiple channels.

1. Introduction

As Web applications spread in almost every domain, novel challenges are posed to developers. The current advances in the communication and network technologies will change the way people interact with Web applications, providing them with different types of mobile devices, for accessing at any time, from anywhere, and with any media services and contents customized to users’ preferences and usage environment. Due to such premises, the traditional design methods will not be anymore exhaustive, since new issues and requirements need to be addressed for supporting multi-channel, context-aware access to services. This paper focuses on such requirements, and proposes some solutions conceived within the WebML method.

WebML (Web Modeling Language) is an already established conceptual model for data-intensive Web applications, which is also accompanied by a development method and a CASE tool [4, 5, 3]. WebML, and in general conceptual modeling, have already proven their effectiveness for the design of personalized Web applications [6, 9]. In this paper we illustrate how some extensions can also support the design of applications able to adapt themselves to the variability of the adopted communication channel and of the usage context.

The paper is organized as follows. Section 2-4 provide an overview of WebML, recalling its basic concepts and notations, and illustrating how the model already supports the specification of one-to-one personalization and multi-channel delivery. Section 5 introduces some new emerging requirements for the design of context-aware applications, and clarifies some assumptions at the basis of our approach. Section 6 and 7 then illustrate the proposed extensions along two design dimensions: data and hypertext modeling. Section 8 presents the extended WebML model at work for the specification of a museum application supporting users guided tours through exposed artworks. Section 9 finally draws some conclusions.

2. WebML: an overview

WebML 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 [4, 5].

The design process starts with the specification of a data schema, expressing the organization of the Web application content. The WebML Data Model adopts the Entity-Relationship (ER) primitives for representing the organization of the application data. Its fundamental elements are therefore entities, defined as containers of data elements, and relationships, defined as semantic connections between entities. Entities have named properties, called attributes, with an associated type. Entities can be organized in generalization hierarchies, and relationships can be restricted by means of cardinality constraints.

The WebML Hypertext Model then allows describing how contents, specified in the data schema, are published into the application hypertext. The overall structure of the hypertext is defined in terms of site views, areas, pages and content units. A site view is a hypertext, designed to address a specific set of requirements. Several site views can be defined on top of the same data schema, for serving the needs
of different user communities, or for arranging the composition of pages to meet the requirements of different access devices like PDAs, smart phones, and similar appliances.

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 and published within pages. In particular, as described in Table 1, content units denote alternative ways to display one or more entity instances extracted from the data schema by means of queries. Also, a specific unit, called entry unit, denotes forms for collecting input values into fields. Unit specification (excepting for the entry unit) requires the definition of a source and a selector: the source is the name of the entity from which the unit’s content is extracted; the selector is a condition, used for retrieving the actual objects of the source entity that contribute to the unit’s content.

Content units and pages are interconnected by links to constitute site views. Links can connect units in a variety of legal configurations, yielding to composite navigation mechanisms. Besides representing user navigation, links between units also specify the transportation of some information (called context) from the source unit to the destination unit.

As reported in Table 2, some WebML units support the specification of content management operations. They allow creating, deleting or modifying an instance of an entity (respectively through the create, delete and modify units), or adding or dropping a relationship between two instances (respectively through the connect and disconnect units).

WebML also provides units for the definition of global parameters. Parameters can be set through the set unit, and consumed within a page through a get unit. The visual representation of such two units is reported in Table 3.

Besides having a visual representation, WebML primitives are provided with an XML-based textual representation, used to specify additional detailed properties, not conveniently expressible in the graphic notation. Web application specifications based on WebML can be therefore represented as visual diagrams, as well as XML documents.

For a more detailed and formal definition of WebML, the reader is referred to [4].

### 3. WebML and one-to-one personalization

Several application requirements impose that users access a sub-set of data and functionality, according to their profile. The WebML method supports personalization of contents and services, recognizing the key principle that users and their roles must be modeled as data. At this aim, the WebML data schema of a Web application typically includes entities representing Users and the Groups in which they are clustered.

Additionally, the delivery of different viewpoints and functions to selected user groups requires designing alternative WebML site views and “attaching” each site view to the user group for which it has been designed. Such an association can be obtained by modeling also site views as data, for example introducing into the data schema a SiteView entity, and connecting it to the Group entity by means of a relationship. In this way, during application execution, it is possible to forward the user to the home page of the appropriate site view, based on the group membership.

The essential aspect of the WebML approach to one-to-one personalization is therefore the inclusion of users, groups and site views as “first-class citizens” in the application data schema. The data schema of every application designed with WebML thus includes the default sub-schema shown in Figure 1:

1. The entity User represents the basic properties of each user; while entity Group represents collective properties.

2. A many-to-many relationship (called Membership)
Table 2. Basic units of the WebML operation model.

<table>
<thead>
<tr>
<th>Unit Name</th>
<th>Visual Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create unit</td>
<td>Create</td>
<td>It specifies the creation of an entity instance.</td>
</tr>
<tr>
<td>Delete unit</td>
<td>Delete</td>
<td>It specifies the deletion of entity instances.</td>
</tr>
<tr>
<td>Modify unit</td>
<td>Modify</td>
<td>It specifies the updating of entity instances.</td>
</tr>
<tr>
<td>Connect unit</td>
<td>Connect</td>
<td>It specifies the creation of a relationship instance.</td>
</tr>
<tr>
<td>Disconnect unit</td>
<td>Disconnect</td>
<td>It specifies the deletion of a relationship instance.</td>
</tr>
</tbody>
</table>

connects User to Group, denoting that users may belong to multiple groups, and that groups cluster multiple users.

3. A many-to-one relationship (called DefaultGroup) connects User to Group, denoting that user may have one group as the default one among groups they belong to. This additional information is useful for assigning users to the default group after they log into the application, and for forwarding them to the site view of their default group.

Figure 1. WebML user model.

The described user model also applies to those Web applications in which users remain anonymous, but are nonetheless individually tracked, for example in virtue of their IP addresses or session identifier. In this case, anonymous users can be considered as instances, although temporary, of the User entity. This permits to provide them with personal data for the duration of the session, for example with an individual trolley in an e-commerce Web application, even though their credentials are unknown.

The definition of relationships between the entity User (or Group) in the user model, and some other information objects in the application content allows modeling preferences over any information object, and ownership over objects created and managed by individual users. For example, in Figure 2 the application entities Artwork and Review are associated to individual users through relationships expressing the user preferences over artworks and user participation in writing reviews over artworks. On this schema it is possible to specify the personalized hypertext of Figure 3. Thanks to the global parameter CurrentUser it is possible to retrieve the current user identifier, and to use it in the selectors of the page units for displaying some personal data (User data unit), the list of user’s preferred artworks (My Artworks index unit), and the list of reviews the user has written about artworks (My Reviews index unit).

Figure 2. Data schema for a personalized Web application.
4. WebML and multi-channel delivery

Site views may also serve the purpose of expressing alternative forms of content presentation on different devices. Each site view may cluster information and services at the granularity most suitable to a particular class of devices.

For example, one application can feature two different hypertexts defined over the same content: one for PC browsers, the other for WAP devices. Given the difficulty of browsing with a small screen, the WML version must be defined so as to provide very concise contents and minimized interactions. Hypertext modeling can be therefore applied to multi-channel applications, by devising the site view structures most suited to the specific delivery medium, that take into account the specific requirements of each device with respect to the amount of information that can be placed within pages and the complexity of navigation.

Once the site view schemas for the different devices are established, producing the actual pages in the proper markup language is a matter of implementation, generally based on the definition of XSL style sheets for the production of markup code.

It is worth noting that WebML requires the definition of distinct site views only when devices show very different rendering capabilities, and therefore require completely different hypertext structures. For devices with similar features, at design time it is possible to model one only hypertext, and then translate it into the multiple mark-up languages required by devices, by applying appropriate XSL transformations over the XML-based site view specification.

5. WebML and context-awareness

Context-awareness is often seen as a recently emerged research field within information technology. From another perspective, it can be however interpreted as just an extension of personalization, addressing not only the users identity and preferences, but also the interaction environment that hosts applications.

Context can be described in terms of properties related to the current user, her/his current activities, the location in which the application is used, the devices, and some other aspects of the environment and of the application itself that can be used for determining the needed adaptation [7, 8]. Besides users data, adaptive solutions for context-aware applications thus exploit context data for providing users with more useful and usable services, in a scenario where, especially due to the mobility and multiplicity of devices, the context can vary.

Context-awareness therefore requires automatisms on the application part that are triggered by changes of anyone of the parameters that make up context.

5.1 Functional requirements

In this paper we focus on WebML and its extensions for supporting the specification of context-aware applications. WebML supports data and hypertext modeling. Thus we will introduce some new concepts along these two dimensions, which allow representing the context status at data level, and also specifying some reactive behaviors of the application with respect to the context state.

Since the WebML modeling approach is data-driven, the use of context within application primarily requires its representation at data level. However, also mechanisms for context data acquisition and access are needed [8]. Therefore, a set of the envisioned extensions allows fulfilling the following goals:

1. Context model representation through a dedicated data sub-schema.
2. Acquisition of context data made available by the usage environment.
3. Continuous update of the context model within the application data source, for reflecting the newly reached context state.

Some other goals are concerned with adaptivity actions over the application hypertext:

1. Adaptivity of currently visited page, based on a new reached context.
2. **Adaptivity of navigation**, through automatic navigation towards pages, within a same site view, which are more appropriate for the reached context.

3. **Adaptivity of the whole hypertext structure**, by means of site views switching, for facing changes of users device, role and activity.

### 5.2 Architectural requirements

In order to reach the previous goals, it is necessary to monitor some context data, and capture them from the environment in which the application is executed. Figure 4 shows the context data flow within a possible architecture tailored to support context-aware hypertext solutions. The two upper layers refer to the application data source and hypertext. The data source includes both the application data (i.e., the business objects that characterize the application domain), and some context data, the latter offering at any moment an updated representation of the context status, which we call context model. The hypertext computation primarily exploits the application data for page and unit computation. We then assume that a subset of hypertext elements, included in the so-called adaptive hypertext, be augmented with adaptive behaviors, and their computation exploit also context data.

As expressed by the gray box, the proposed architecture also features context-sensing mechanisms for capturing context data from the application execution environment. Since context sensing is a technological issue, which does not affect the specification of the final hypertext, we will not focus on it. We only assume that some solutions allow sensing such data, that become available to the application in three modalities:

- Parameters generated at client-side are sent back to the application in append to the URL of requested pages;
- The values of some session parameters, managed at server-side, are set according to newly captured context data;
- The sensing mechanism update the context model at data level, for example through asynchronous services. The new stored data are then used for the computation of the adaptive hypertext.

It is worth noting that although the first two acquisition mechanisms act at the hypertext level, some operations in the adaptive hypertext have the role of keeping the context model updated with respect to the fresh captured values.

The reminder of the paper describes the WebML extensions that are required to support context awareness. Similarly to other extensions proposed for WebML (e.g., for process management [2] or for Web services [1]), the novel concepts refer to (i) the definition of a suitable data schema for modeling context (Section 6), and to (ii) the definition of additional hypertext units for managing adaptivity (Section 7).

### 6 Modeling context through data sub-schemas

The modeling approach proposed by WebML is data-driven. We therefore propose to enrich the application data schema with a context model, that is the representation as entities of metadata needed for supporting adaptivity. By explicitly representing these concepts within the data schema, many useful customization policies can be expressed declaratively in the hypertext schema, instead of being buried in the source code of the application.

The context model can vary depending on the application domain, and also on the adaptivity goals to be fulfilled. For this reason, we will not provide a precise, rigid characterization of context. Rather, we will introduce some guidelines about how to extend the data schema through context metadata.

Figure 5 shows a possible sub-schema for representing context information, which extends the user model sub-schema illustrated in Section 3. It is possible to note two main extensions: some context entities, for representing context data, and a new relationship between the entities Group and SiteView.

The context entities are associated to each user. In Figure 5, they for example represent the user’s device, location, and activity. Some applications may also require a different set of entities for representing the context. However, our approach just prescribes to have such entities associated (directly or indirectly) to the User, which is the starting point
for navigating within the context model and extracting context information.

The new many-to-many relationship between Group and SiteView represents all the site views associated to a given group, among which users can switch during the application execution, according to the adaptivity rules defined in correspondence of context changes. With respect to the WebML user model described in Section 3, this new relationship expresses that users belonging to a given group can act in different contexts and hence may require different site views to switch among, as soon as the context changes.

Site view switching can therefore occur when the user change devices, thus a new hypertext structure is required for accessing contents - this allows capturing multi-channel requirements. Also, site view switching is needed when contents and services to be provided in a new context are totally different, thus a new view over the application data and functions is required. For example, a museum application could provide users with completely different hypertexts when they are visiting artworks inside the museum building using a PDA for accessing artwork description, and when they are accessing the application outside the building, even with the same device, for getting information about the current expositions and the opening hours.

The one-to-many relationship remains unchanged with respect to the WebML user model. In the context model it however represents the association of a group with a default site view, to be provided to a group of users when information about the current context are not available.

7 Extending the hypertext model for capturing context awareness

Besides introducing a data sub-schema for representing context, it is necessary to introduce some extensions also at hypertext level, for specifying operations for context model updating and hypertext adaptivity.

Our basic assumption is that context awareness is a property to be associated only to some container elements, i.e., pages, areas, and site views. The contained elements will inherit the adaptivity actions defined for their containers and, additionally, can be associated with more specific actions.

As represented in Figure 6, it is possible that only some pages, areas, and site views need to be adaptive with respect to context. For example, some “access” pages, as those reported in Figure 6 for invoking searching services, might not be affected by the context of use and therefore do not need adaptivity actions.

During requirements specification, the application designer will identify the context-aware elements and tag them with a “C” label (standing for context-aware). The label indicates that:

- The marked pages are augmented with refresh mechanisms that periodically generate page requests.
- Some adaptivity actions are associated with the marked pages, and during the application execution, such actions must be evaluated and applied every time a refresh request is generated for the page.

The refresh mechanism and the execution of the adaptivity actions allow capturing and reacting to context changes that can occur while users are visiting a “C” page.

During the execution of the application, the process of context-based adaptivity is therefore initiated by the request of a “C-labeled” page. When the page is entered for the first time, the user selection is used for computing the page contents to be displayed. At periodic intervals of time, the refresh event then triggers the acquisition of context information first, according to the mechanisms already described in Section 5.2. If fresh context data are captured as session parameters, some operations for updating the context model within the application data source are performed. Additionally, some operation chains allow retrieving possible data
from the data source, or also computing new data, needed for adapting the page to the renovated context status. Only after such actions, the page computation starts.

The central context-aware element is therefore the page. As shown in Figure 6, we however propose to define “C” areas as grouping facilities, that allow us to insulate some redundant adaptivity actions to be performed for every “C” page within the area, and represent them once at the area level. The adaptivity actions associated to areas are to be executed every time a “C” page within the area is accessed, and before executing the adaptivity actions specified at the page level. Typical actions at the area level are the acquisition of fresh context data and the consequent updating of the context model. We therefore propose two levels for the specification of context adaptivity actions:

1. Actions for context model management, addressing operations for context data acquisition and context model updating. Such actions can be associated with the most external containers (site views or areas), and are inherited by all the internal elements (pages or areas).

2. Actions for hypertext adaptivity, addressing the definition of rules for page and navigation customization. Such actions are generally associated with “C” pages.

The main profit of extending the “C” property to areas and site views is the reduction of specification redundancy. Therefore, in general we assume that each request of a “C” page causes first the computation of actions associated to its “C” containers (if any), starting from the most external one. In the following sections, we will introduce some new hypertext constructs that allow us to express actions for context-model management and hypertext adaptivity.

7.1 Units for accessing fresh context data

In order to specify actions for accessing fresh context data, according with the three access mechanisms illustrated in Section 5.1, we use three units able to access both sensed data and database objects.

The Get unit, already described in Section 2, is used for accessing session parameters.

The Get URLParam unit (see Figure 7) allows retrieving user-device generated parameters, appended to the request of the current page. WebML already supports the specification of link parameters and global parameters that at runtime correspond to values appended to the page URL. However, those parameters are server-generated, while through the Get URLParam unit we intend to specify parameters generated by client-side sensing mechanisms.

Finally, the Get Data unit (see Figure 8) is used for extracting values (both scalar values and values set) from the data source, according to a selector condition. It is similar to the data unit, with the only difference that the retrieved data are not published in a page, but just used as input to some successive operations. It is therefore a facility for accessing database information whenever no visualization is needed, e.g. in situations where certain data are just used to evaluate condition expressions.

7.2 Units for condition evaluation

The execution of context model updating, as well as of hypertext adaptivity actions depends very often on the evaluation of some conditions. At this proposal, in order to specify test over certain conditions, we use two control constructs, the If and the Switch operations (see Figure 9), that have been recently proposed for extending WebML for modeling Workflows [2].

The two units have incoming links carrying possible parameters, and use such parameters for evaluating an expression. If the evaluation succeeds, one of the output OK links is followed, depending on the result of the evaluation. In particular, the If unit evaluates a Boolean expression, and provides two different OK links, one to be followed when the condition is true, the other when the condition is false. The Switch unit evaluates an expression, defined over the input parameters, and provides a set of OK links, to be followed depending on the matching between the results of the expression and the guard condition specified over the link.

If the expressions for the two units are not computable, the KO link is followed. In addition, the switch unit presents one OK link without guard condition, which is followed...
when the computation results do not match any of the guard conditions, but the calculation has been successful.

### 7.3 Unit for site view switching

In some cases, the change of context may ask for a change of site view. In order to specify this action, we have introduced the ChangeSiteView unit (see Figure 10), which takes in input the identifier of the new site view and of the specific target page within it. The input link also transports all the parameters required by the target page for reconstructing the current state, as well as those parameters representing selections operated by the user while interacting with the origin site view, and that can be useful for the computation of the new page.

### 8 Case study: the museum guided tour

In this section, we show by means of a simple example how to integrate application data and context data for the provision of context-aware hypertexts.

Our scenario refers to an application assisting visitors of a museum. It can be seen as a new enlargement of an already existing Web application, publishing general information about the current and future expositions, its artworks and its artists, and providing the user with a reservation mechanism for the various guided tours through the museum. The desired extension is a guided indoor visit. It allows users to walk along the museum rooms, accessing the description of exposed artworks through a portable device, for example a PDA, able to access the wireless network inside the museum.

For location detection, we suppose the device being able to sense its position and adding the result as parameters to each page request. The device is able to gather the signals that the numerous transmitters installed above the exposed artworks emit, which allows locating the visitors position inside the museum and delivering the associated contents. The positioning of the transmitters forms sensitive cells within the rooms (see Figure 11) that the receiver is able to identify and communicate back to the application via a URL-parameter that carries the unique identifier of the corresponding area. Whenever the visitor enters a new cell, the application shows the description of the corresponding artwork or artist or, when the user is in the center of the room, it shows a general description of that room.

### 8.1 Data modeling

As reported in Figure 12, the data schema related to the indoor visit application supports the delivery of data concerning artworks, artistic movements and artists biographies.

The bottom of the figure is about the Context Model. In this case, for sake of simplicity, beside entities representing the user model, only the Location entity is used for completing the context representation. Its attributes reflect the specific features of the museum infrastructure previously described. Its instances represent all the sensible areas for
Figure 12. Entity-Relationship schema of the museum application, and identification of Application Data and Context Data.

which the application is able to supply descriptions. In a given moment, a user can be associated to a specific location. If the user location changes, then the relationship between User and Location gets updated.

The connection point between context data and application data is the relationship between sensitive areas inside the rooms and the exposed artworks. The relationship expresses that each area exposes always exactly one artwork, while not every artwork is placed in an area inside the museum; the dataset may indeed contain artworks that are not currently exposed to visitors. Connected to the Artwork entity, we can then find the remaining application data, modeled by means of the entities Artist and Movement.

8.2 Hypertext modeling

Figure 13 reports a fragment of the hypertext schema for the museum application, which will allow us to show the new introduced extensions at work.

The hypertext represents a context-aware area (as represented by the “C” label at the right upper corner), which includes two context-aware pages: one presenting details of artworks exposed in the current user location, the other showing details about the current room.

The “C” property associated to the area triggers an action chain, for the retrieval of fresh context parameters and the consequent context model updating, whenever one of its “C-labeled” pages is accessed through a refresh request. The chain starts with the retrieval of the current user identifier (Get User unit), then of the location currently associated to the user (Get Location unit), and finally of the (possibly new) user location sensed through the device (Get URLParam unit).

The user location stored in the data source and the new sensed location are used by the If unit, for evaluating if the two values are different. If yes, the context model needs to be updated. Therefore:

1. A Disconnect operation deletes the current relationship between the user and the currently stored location;

2. A Connect operation creates the new relationship between the user and the location instance corresponding to the new sensed value.

As actions associated to the context-aware pages, the figure shows some operations for customizing the content of the current page to the current user location.

The chain of Get units, represented in the top of the figure, extracts from the database the artwork associated to the users current position, and provides the following If unit with this value. The If unit checks whether there exists an artwork exposed within that position and, in case the artwork has been found, updates the data unit of the page Artwork Details. If there is no artwork associated to
the current user position, it means that the user is located in the center of the current room. Therefore, the If unit trig-
gers a page change by first extracting the respective room identifier from the database, and then forwarding the user to the page Room Details by means of a contextual link carrying the retrieved identifier. This action implies the pre-
sentation of the page Room Details in place of the page Artwork Details, i.e., it represents an adaptivity action over navigation.

The page Room Details is aimed at presenting a de-
scription of the room, plus the index of its exposed art-
works. Its adaptive actions are similar to those defined for the Artwork Details page, and cause a page updating with new contents, or a navigation towards the Artwork Details page.

9 Conclusion

This paper has presented some issues for the design of multi-channel context-aware applications, and has proposed some solutions for their conceptual modeling. Several approaches have been proposed for the development of such class of applications (see [7] for a survey). However, since only very few methods are model-based, often customization results into programming scripts buried within the application code.

The solution we have proposed is based on the adoption of the WebML conceptual model, and consists of the intro-
duction of a new modeling dimension, for specifying the context model, as well as the required operations for context model management and hypertext customization. It offers the advantage, which is proper of conceptual modeling, to reason at a high level of abstractions, without being influ-
enced too much by implementation issues. Also for this reason, we believe that, although proposed within WebML, our solutions can be easily adopted within other models or methods for Web applications development.

Our current work focuses on formally defining the seman-
tics of the new introduced constructs, and on identifying possible changes to hypertext computation logic the extensions require for. We are also studying some synchroniza-
tion issues that might arise in case of parallel sessions, through different devices, by the same user.

In the continuation of the MAIS project, which support this research, we aim at studying multi-modal applications, i.e., applications which take place by synchronizing the de-
ivery of coordinated contents over two or more media; we will probably consider first visual and audio combinations. This research includes classical aspects of multi-media de-
ivery (such as synchronization on various devices) but with the addition that each delivery can be considered as a Web application in its own rights, e.g., with browsable user inter-
faces (which, in the case of audio channels, may correspond to receiving explicit commands from the users and sending back vocal messages). Adaptation in this case may affect one or both of the channels (e.g., amplify one of them when the other one cannot be properly used due to changes to the context). The modelers will in this case require not only the adaptation of individual site views, but also the coordination among multiple site views.

On a separate ground, we will consider how the exten-
sions which are proposed in this paper, and currently tested by "encoding" adaptability in a conventional Web application, can be implemented on WebRatio, the current WebML case tool [3]; this will entail both a modification of the site designer component, to enable the model extensions, and of the automatic generation of the application from the speci-
fications. Both changes are not particularly hard, due to the extensibility of the WebRatio architecture.

10 Acknowledgements

We are grateful to the WebML team for the useful dis-
cussions and the valuable suggestions.

This research work is founded by the MAIS (Multi-
channel Adaptive Information Systems) FIRB project.

References

[8] A. Kobsa, J. Koenemann, and W. Pohl. Personalized Hyper-
media Presentation Techniques for Improving Online Cus-