An XML-Based Architecture for Adaptive Web Hypermedia Systems Using a Probabilistic User Model

Mario Cannataro
ISI-CNR, Via P. Bucci, 41/c
87036 Rende, Italy
cannataro@si.deis.unical.it

Andrea Pugliese
ISI-CNR, Via P. Bucci, 41/c
87036 Rende, Italy
andreapugliese@tin.it

Abstract

Web-based hypermedia systems are becoming increasingly popular as tools for user-driven access to information and services. This paper presents an architecture for the development of web-based Adaptive Hypermedia Systems. The architecture uses weighted graphs of XML documents to describe the application domain and a probabilistic model to adapt the web-site content generation and presentation to the user's behaviour. The user's behaviour is modelled using a probabilistic model and the most promising profile, that is a "view" over the application domain, is dynamically assigned to the user, using a discrete probability density function.

1. Introduction

Web-based hypermedia systems are becoming increasingly popular as tools for user-driven access to information and services. The linking mechanism of hypermedia offers users a large amount of navigational freedom so that it becomes necessary to offer support during navigation. Moreover the classes of users to be dealt with are becoming increasingly heterogeneous and demanding due to the worldwide deployment of applications. Typical application fields for hypermedia systems are on-line learning and teaching, electronic commerce, on-line advertising.

To efficiently allow the realisation of user-adaptable contents and presentations, a clear separation between basic multimedia contents, domain model and user model should be achieved. Moreover, methodologies to track in a non-invasive way the user's behaviour and to mine the access data should be applied.

Basic components of Adaptive Hypermedia Systems are:

- the Application domain model;
- the User model;
- the Adaptation of presentations with respect to the user's behaviour.

The Application domain model is used to describe the hypermedia basic contents and their organisation to depict more abstract concepts. The User model attempts to describe the user's characteristics and preferences and his/her expectation in the browsing of hypermedia. The adaptation of the Application domain presentation to the User model follows various schemes that can be summarised as follows [9]:

- In *adaptable* hypermedia the user can provide some profile, and the contents are delivered with respect to the selected profile;
- In *adaptive* hypermedia the system monitors the user's behaviour and adapts (chooses) the presentation accordingly. The user's behaviour is often deducted from his/her browsing activity;
- In *dynamic* hypermedia there are no predefined presentations, so on the basis of the user's behaviour, the system generates presentations combining in a dynamic way atomic components, such as images, texts, sounds and so on.

This paper presents an architecture for the development of web-based Adaptive Hypermedia Systems (AHS), using weighted graphs of XML documents to describe the application domain and a probabilistic model to adapt the web-site content generation and presentation to the user's behaviour. XML data-centric orientation makes it possible to elegantly describe application domain, data access and dynamic data composition functions, thus resulting in an open architecture, allowing the use of pre-existing multimedia basic data (e.g. stored in relational databases).

The application domain, represented by a set of weighted graphs where the nodes are XML documents and the arcs are links between them, is regarded using different predefined user profiles. For each profile, the arc's weight represents the probability to follow the corresponding link (and to reach the next node).

During the browsing the user's behaviour is monitored probing a set of variables and a probabilistic model is constructed. So dynamically, on the basis of this probabilistic user model, the system attempts to assign the user to the "best" profile. The next page presented to a user is then generated with respect to the assigned profile.

The rest of the paper is organised as follows. Section 2 recalls related work. Section 3 summarises characteristics of XML useful to manipulate data in a dynamic way.

0-7695-0789-1/00 $10.00 © 2000 IEEE

Section 4 describes a general approach to model application domain. Sections 5 and 6 present a probabilistic approach to model the application domain and the user's behaviour, and a method to dynamically assign a user to an available profile. Section 7 depicts a multi-tier architecture to design web-based AHS. Finally, Section 8 contains conclusions and outlines future work.

2. Related work

In the last years some frameworks to develop Adaptive Hypermedia Systems have been developed. As said before, basic components of an AHS are the Application domain model, the User model and the methods to adapt presentations to the user.

The most promising approach in modelling the Application domain is data-centric, and researches are made to adapt well known modelling techniques used in databases to hypermedia systems [1, 8]. Very often Application domain data are organised on three levels distinguishing among [11]:

- *Information fragments* (or *atomic concepts*), at the lowest level, like texts, sounds, images, drawings, etc;
- *Presentation units* (or *pages*), composed of fragments, that are presented to the user;
- *Abstract concepts,* at the highest level, representing larger units of information. Abstract concepts and Presentation units can be organised as graphs.

User models (*profiles*) describe a set of user's characteristics (*overlay models*), typically represented by a set of *attribute-value* pairs, or indicate the user's belonging to a group (*stereotype models*) [6]. In the construction of the profiles, many different aspects are generally taken in account: user's *knowledge* of the Application Domain to deal with, *goals* and *preferences* that guide his/her browsing, or the *reading key* the user wants to obtain from the hypermedia.

Profile updating (in the case of overlay models) or an assignment of the user to a profile (in the case of stereotype models) must be made dealing with some information the user "shares" with the system [18]. This information can be collected directly involving the user (*co-operative user modelling*) or just observing his/her actions (*automatic user modelling*). Automatic user modelling is obviously not completely reliable, so it can be useful to let the user change his/her profile "on the fly", giving him/her a hint based on automatic evaluations if necessary.

Presentation adaptation can be generally distinguished into *adaptive presentation* and *adaptive navigation support*. Adaptive presentation is a manipulation of fragments: they can be conditionally included, can be made "less visible" in the pages or their order can be changed. Another interesting technique is based on the use of *stretchtexts*, e.g. text fragments that can be also presented as short placeholders; the initial presentation of fragments is decided by the system while the user can "stretch" or "shrink" fragments on the basis of his/her interests.

Adaptive navigation support is a manipulation of links: a "direct guidance" (e.g. a "next" button) can be featured, links can be conditionally included, sorted, disabled, or annotated on the basis of their importance; their structure can be presented providing an *adapted map* of the hypermedia.

The presentation units can also be adapted accordingly to some *context variables* (e.g. browsing time or user's location), or on the basis of the behaviour of many users. As an example, relying on the *visit-coherence* assumption a link to a node *n2* can be included into node *n1* if many users have reached *n2*, indirectly, starting from *n1* in the past [21]. Moreover, some clusters of related pages can be created, and they can be made accessible directly by an index page [22].

Some recent adaptive hypermedia systems are outlined below:

- The *AHA* system [10] uses overlay profiles with Boolean attributes, which are updated presenting tests to the user (for tracking *knows-about-something* concept) or logging the pages he/she has read (for tracking *read-about-something* concept). The system conditionally includes fragments using Boolean values, and supports link annotation by means of link classes that are presented into different colours.
- The *Torii* system [8] dynamically assigns a stereotype profile to the user with *Event-Condition-Action* rules, on the basis of his/her behaviour; *site views* corresponding to stereotypes are presented.
- The *AVANTI* project [14] acquires *primary assumptions* about the user with an initial interview and observing certain dialog actions (e.g. a request of an explanation); based on them draws inferences on further assumptions.
- In [2] a set of probabilistic rules is used to assign an initial stereotype profile to the user, on the basis of a questionnaire.
- The *SETA* system [3] constructs adaptive hypermedia useful to guide the user in the exploration of a product catalogue in a virtual store, on the basis of his/her overlay profile.
- In [15] fragments are greyed out if they are not useful at the moment.
- *InterBook* [12], an education-oriented system, supports link annotation changing icons and fonts of the text anchor on the basis of the user's knowledge.
- *MetaDoc* [5] uses stretchtext technique to present personalised views and to track the user's interests.
- The *Strudel* system [13] applies a *site-definition query* to the underlying data to define the web-site structure, thus resulting in a "site graph", which represents both the site's content and structure. The author specifies the

visual presentation of pages in Strudel's HTML-template language.

- The *Araneus Data Model* [4] is inspired to the typical web-site structures. On its basis, a language called *Ulixes* is used to build database views of the Web, and a language called *Penelope* is used to define hypertexts from relational views.
- *SelectCast* [17] is a commercial tool oriented to electronic commerce that uses neural networks to analyse the user's behaviour and predict his/her interests, using a *virtual salesman* metaphor.
- *Active Views* [1] is a system oriented to electronic commerce which offers a declarative view specification language to describe a web application that allows users to work interactively on the specified data in a standard distributed environment.

3. Using XML to describe adaptive contents

XML (*eXtensible Markup Language*) is a markup language useful for the construction of structured documents [26]. Unlike HTML that has been designed to be *display-centric*, XML is *data-centric*: it expresses the structure and eventually the meaning of the data contained in a document leaving its visualisation to subsequent elaboration. XML specifications do not fix a tag-set, leaving the possibility of defining *classes* of documents; this feature makes XML a meta-language for the definition of markup languages. XML is a subset of SGML (*Standard Generalized Markup Language*) and it can be seen as a synthesis between the simplicity of HTML, adding extensibility to it, and the expressive power of SGML, with less complexity [23].

Document classes are defined by DTDs (*Document Type Definitions*), definitions of some constraints about sequence and nesting of tags and about allowed attributes for each tag (with their values if necessary). *Valid* documents respect the definitions given in a DTD, while *Well-Formed* documents respect only the base syntax of XML. A valid document is obviously also Well-Formed. Several *XML Parsers* are already available, public domain in most cases, which perform parsing of the documents and optionally their validation. An XML parser after having verified the absence of errors in the document, generates a representation of it accessible by one of the two most widespread APIs:

- *SAX* (*Simple API for XML*) that generates a series of *parsing events* notified to the objects that deal with the document [20];
- DOM (*Document Object Model*) that builds up an in-memory tree representation of the document and allows its browsing and elaboration [25].

As an example, consider the following DTD, which defines a simple structure for an e-mail message:

```
<!ELEMENT mail (Date,Subject,Textbody)>
<!ATTLIST mail
    Recipient CDATA #REQUIRED
    Sender CDATA #REQUIRED>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT Subject (#PCDATA)>
<!ELEMENT Textbody (P)*>
<!ELEMENT P (#PCDATA|Name)*>
<!ELEMENT Name (#PCDATA)>
```

That DTD indicates *mail* as the document root element, with two required character data attributes, *Recipient* and *Sender*, and three required sub-elements, *Date*, *Subject* and *Textbody*. *Date* and *Subject* are parseable character data elements while *Textbody* can optionally contain *P* sub-elements and so on. A valid XML document with respect to that DTD is:

```
<?xml version="1.0"?>
<!DOCTYPE mail SYSTEM "mail.dtd">
<mail Recipient=
    "cannatar@si.deis.unical.it"
    Sender="andreapugliese@tin.it">
<Date>03-03-2000</Date>
<Subject>Java literature</Subject>
<Textbody>
    <P>Hello Mr <Name>Cannataro</Name>,</P>
    <P>Please read <Name>Cornell</Name>'s
        text</P>
    <P>"Core Java vol.2"</P>
    <P>Best wishes</P>
</Textbody>
</mail>.
```

A DOM parser would build the in-memory tree shown in figure 1.

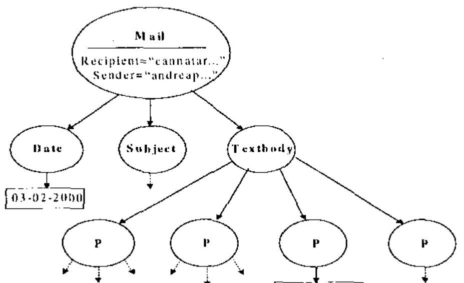

Figure 1. DOM representation of a XML document.

Visualisation of XML documents can be performed directly as trees or transforming them in a visualisation format (HTML, TeX, RTF, PostScript). Transformations can be carried out through *Cascading Style Sheets* [24], or by using the more expressive *eXtensible Stylesheet Language* (XSL) [27]. The most flexible solutions are based on direct use of SAX or DOM APIs, with a completely controlled transformation of documents.

3.1. Use of XML for the dynamic manipulation of data

XML distinguishes among content, structure and style and is therefore very well suited for the description of data on which to perform dynamic manipulations. It can be useful in the description of concepts, of relationships among them, of presentation units and of their way of being dynamically composed. It allows to remain quite abstract with respect to the variety of information resources on which the hypermedia is built (e.g. allowing the extraction of data from heterogeneous databases) and with respect to the various possible visualisation modes. A representation of the hypermedia can in fact be based on meta-informations about its content [21] while data coherence is guaranteed by precise definitions contained in DTDs. XML can also be useful in the description of rules for profile definition and updating [8].

In the case of "pure" *stereotype profiles*, presentation units can be represented as XML documents, which are to be instantiated with respect to the user's profile, viewed as a parameter in the document. A presentation unit can be made up of two sections, *fragments* and *content*. The fragments section contains a set of elements indicating fragments that will be extracted from the database to construct the page and defines some aliases for them. The content section describes how this fragment must be presented to the user, referring to the declarations made into the fragments section.

As an example, let us consider the generation of an adaptive presentation in the field of Cultural Assets, where we suppose the existence of three user profiles (*generic*, *tourist* and *expert*). The relational table *img_table(key, bitmap, place, ...)*, storing images and their attributes, is used as source of fragments.

Then the XML description of a presentation unit that shows an image extracted from this table could have the following structure:

```
<page>
<title>Cool image</title>
<fragments>
<table name="img_table">
    <tuple profile="expert"
        key="13" alias="img1">
        <fragment
            attribute_name="bitmap"
            type="image"/>
        <fragment
            attribute_name="place"
            type="text"/>
    </tuple>
    <tuple profile="all"
        key="7" alias="img2">
        <fragment
            attribute_name="bitmap"
            type="image"/>
    </tuple>
<table>
</fragments>
<content>
<image_fragment profile="tourist"
    alias="img2.bitmap"/>
<image_fragment profile="generic"
    alias="img2.bitmap"/>
<image_fragment profile="expert"
    alias="img1.bitmap"/>
<text profile="expert">
    <normal> This image refers to
    </normal>
    <bold> <text_fragment
        alias="img1.place"/>
    </bold>
</text>
</content>
</page>
```

If the "Cool image" XML document (presentation unit) is instantiated for the expert profile, the system will show the JPEG image contained in the "bitmap" column of the tuple with key "13". Moreover, a short text explaining the place the image refers to, which is extracted from the same tuple in the "place" column, is also shown. In the case of generic or tourist profile, the system will only show the JPEG image contained in the "bitmap" column of the tuple with key "7".

In this example, a DTD for the page could have the following structure:

```
<!ELEMENT page (title?,fragments?,content)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT fragments (table+)>
<!ELEMENT table (tuple+)>
<!ATTLIST table name CDATA #required>
<!ELEMENT tuple (fragment+)>
<!ATTLIST tuple profile
    (generic|tourist|expert|all) #REQUIRED
    key CDATA #REQUIRED
    alias CDATA #REQUIRED>
<!ELEMENT fragment EMPTY>
<!ATTLIST fragment
    attribute_name CDATA #REQUIRED
    type CDATA #REQUIRED>
<!ELEMENT content (image_fragment|text)+>
<!ELEMENT image_fragment EMPTY>
<!ATTLIST image_fragment profile
    (generic|tourist|expert|all) #REQUIRED
    alias CDATA #REQUIRED>
<!ELEMENT text (normal|bold)+>
<!ATTLIST text profile
    (generic|tourist|expert|all) #REQUIRED>
<!ELEMENT normal (#PCDATA|text_fragment)+>
<!ELEMENT bold (#PCDATA|text_fragment)+>
<!ELEMENT text_fragment EMPTY>
<!ATTLIST text_fragment
    alias CDATA #REQUIRED>
```

In the case of *overlay profiles*, an XML-based data description can also be used to implement dynamic personalisation. XML can in fact describe *rules* through which fragments have to be extracted and combined. The Event-Condition-Action rules are performed on the basis of some *attribute-value* pairs contained in the user profiles, that are dynamically updated by the system.

4. Application domain modeling

This Section presents a general model to describe Application Domain using XML documents. The main goal is to explain the general guidelines to describe application data in such a way the domain model, the user model and the adaptation model can be made orthogonal. In our system the modelling of the Application domain data is organised on different levels as outlined before:

- *Information Fragments* (or *atomic concepts*), at the lowest level, like texts, sounds, images, drawings, etc; The information fragments are stored in relational databases or in files. In fact, very often these are pre-existing data and it could not be suitable to convert their formats.
- *Presentation Descriptions* composed by XML documents stored in a XML repository (a relational database or a file system). A presentation description describes which information fragments may occur in a presentation unit, and how these data can be selected on the basis of different parameters, such as user profile, user's behaviour and local context. XML documents comprise the *fragments* and *content* sections (see Section 3), and a set of links to other documents, that are used to represent relationships or simply the next documents that should be visited. The links are annotated by a weight that can represent their importance with respect to each other.
- *Presentation Units* (or *Pages*) composed of fragments that are presented to the user. They are obtained evaluating conditions on variables expressing user's behaviour or local context and instantiating Presentation Descriptions. The pages are dynamically generated (e.g. using DOM APIs) in a target language depending on the user terminal (XML, HTML, WML).
- *Elementary Abstract Concepts*, representing larger units of information, with a proper semantic. An Elementary Abstract Concept is composed by one or more Presentation Descriptions organised in a graph, which is assumed to be acyclic for simplicity. Arcs represent relationships between elementary concepts or navigation requirements, e.g. to learn an abstract concept a sequence of elementary concepts must be learned. In simple cases each Presentation Description could represent an abstract concept.
- *Application Domain*. Finally, an Application Domain is composed by a set of concepts organised in a graph. Arcs represent relationships between Abstract concepts.

The proposed model supports *stereotype profiles*. An Application Domain is organised in such a way it is possible to regard it along different views corresponding to stereotype profiles. A user belonging to a given profile will see the corresponding view of the domain.

If a Presentation Description occurs in more profiles, than the corresponding XML document (from here named node) will contain all the corresponding fragments, contents and links. Naturally, the same concept can be represented differently with respect to profiles, referencing either different fragments or links: a single node (or a group of them) represents the same abstract concept with respect to different profiles.

With respect to the adaptation of presentation descriptions to pages the model supports both *adaptive presentation* and *adaptive navigation*. To support *adaptive navigation* the model uses a weighted graph-based representation of XML nodes.

An Application Domain D, with M different profiles (reading keys), is a set of N XML documents where the generic document i ($i=1,\dots,N$) contains, for each profile $k=1,\dots,M$, L_{ik} annotated outgoing links. It can be mapped as a multigraph where each node correspond to a XML document and each directed arc to an outgoing link:

$$G = (N, \{L_{i,k}, i = 1,\dots,N, k = 1,\dots,M\})$$

For the sake of simplicity, in the next sections we will refer to the multigraph G as the set of the directed graphs G_k, $k=1,\dots,M$, obtained extracting from G the nodes and arcs corresponding to each profile. Each G_k is named *Logical Navigation Graph*.

$$G_k = (N_k, E_k), \qquad N_k \subseteq N, E_k = \bigcup_{i \in N_k} L_{i,k}$$

The generation of a presentation unit from a XML document i, for the profile k, runs as follows:

1. The graph node i (i.e. the XML document) is loaded.
2. The data portion of the presentation unit is generated on the basis of the profile k and eventually considering context variables.
3. The link portion of the presentation unit is generated considering the weights of the corresponding graph arcs. The links relevant to the profile k can be ordered on the basis of weights, or a threshold function could be applied to hide "light" weight links.

To summarise, our Application Domain model:

- uses a *layered* structure of data,
- supports stereotype profiles and adaptive presentation,
- uses annotated links to support adaptive navigation.

It should be noted that the previous description is intentionally general. For example, we do not detail the nature of annotated links. The following sections will use this general model as a basis to develop a probabilistic domain model and user model, and the corresponding adaptation algorithm.

5. The probabilistic application domain model

Let $G_k=(N_k,E_k)$, where $k=1, \dots, M$, $N_k \subseteq N$ and E_k is a set of arcs (i,j) with i and j belonging to N_k, be the set of M Logical Navigation Graphs describing an Application Domain model, the Probabilistic Application Domain model is obtained adding the following weights to each arc.

The weight $W_k(i,j)$ of the arc (i,j) in the graph G_k is the conditional probability $P(j|k,i)$, namely the probability that a user belonging to the profile k follows the link to the j node having already reached the i node:

$$W_k : E_k \rightarrow [0,1]$$

$$W_k(i,j)= P(j|k,i),\ (i,j)\in E_k,\ k=1, ..., M$$

$P(i|k,i)$ is considered to be always zero, as it is impossible a link from a node to itself. For each node i, the sum of the weights of outgoing arcs, for each profile, is always one.

$$\sum_{(i,j)\in E_k} W_k(i,j) = 1, \quad k = 1,...,M$$

A path S in G_k is defined as an ordered set of nodes:

$$S = \{s_0, s_1, \ldots, s_l : (s_j, s_{j+1}) \in E_k, j = 0,\ldots,l-1\}.$$

We do not use the standard arc-based definition of a path because relaxing the condition $(s_j, s_{j+1})\in E_k$ allows to consider a path involving different Logical Navigation Graphs. This could happen if a user in the profile k chooses a link from a node s_j, to a node s_{j+1} and he/she is moved to a new profile h.

The probability that a user belonging to the profile k follows the S path is:

$$P_S^k = \prod_{j=0..l-1} W_k(s_j, s_{j+1})$$

so P_S^k is the product of the probabilities associated to the arcs belonging to the S path. A high value of P_S^k indicates that the visited nodes in S are relevant to the profile k.

The "shortest" path $\tilde{S}_{ij}^k$ between two nodes i and j for a given profile k is the path with the maximum joint probability given as:

$$\tilde{P}_{ij}^k = \max_{S_{ij}^k} (P_{S_{ij}^k}^k)$$

where S_{ij}^k is the generic path between the nodes i and j through arcs belonging to the profile k. The "shortest" path for each profile can be computed once.

Given a set of visited nodes, the distribution D shows how the visited nodes are distributed with respect to each profile. For example, let (n_1, n_2, n_3) be the visited nodes: if node n_1 belongs to profiles k_1 and k_2, node n_2 belongs to k_2 and k_3 and node n_3 belongs to k_1 and k_4, the distribution can be evaluated as $D=[(k_1, 2), (k_2, 2), (k_3, 1), (k_4, 1)]$.

In summary, the modelling of an application domain consists of:

- defining the M profiles through which to view the site contents;
- composing the XML documents and related links, with respect to the profiles;
- composing the M Logical Navigation Graphs assigning probabilities to arcs.

6. The probabilistic user model

The Probabilistic User model collects information about the user's behaviour to build a discrete probability density function $A(k)$, with $k=1, ..., M$, measuring his/her "belonging probability" to each profile (i.e. how much each profile fits him/her).

The user's behaviour is stored as a set of *attribute-value* pairs. The main attributes are:

- the current profile;
- the discrete probability density function $A(k)$, $k=1,...,M$, measuring the user's "belonging probability" to each profile;
- the recently followed path $R = [R_1, ..., R_{r-1}, R_r]$, which contains the last visited nodes, where R_{r-1} is the current node and R_r is the next node. The last arc (R_{r-1}, R_r) is the outgoing link chosen by the user.

Browsing starts from the presentation unit associated to a starting node H; the user is assigned to a generic profile or to one calculated on the basis of a questionnaire. The value of $A(k)$ is one for that profile and zero for the other ones. During the user's browsing activity $A(k)$ is updated by the system and the user's profile is changed consequently. So, on the basis of the user's behaviour, the system dynamically attempts to assign the user to the "best" profile.

When a user visiting the node R_{r-1} requests to follow a link, the system has to decide the (new) profile for him/her and generate the page corresponding to the R_r node in the computed profile. On the basis of the previous attributes, the system computes the new probability density function, $A'(k)$, then it decides the (new) profile to be assigned to the user. To avoid continuous profile changing it is possible to keep a profile for a given duration (i.e. the number of traversed links), evaluating the $A'(k)$ distribution at fixed intervals. Moreover, the user can also "force" the changing of his/her profile.

The new profile could be chosen making a random extraction over the $A'(k)$ distribution or referring the highest $A'(k)$ value. Besides, other than assigning the user to a new profile, the system could support the dynamic composition of fragments referring to more profiles, if the distribution presents high values for those profiles.

6.1. The evaluation of the belonging probabilities

The algorithm to compute *A'(k)* on the basis of the user's actions has the following structure.

Input:
- The discrete probability density function *A(k);*
- The recently followed path $R = \{R_1, \ldots, R_{r-1}, R_r\}$, composed by the last *r* nodes visited by the user;
- The length of *R*, *r*, that is a parameter of the system.

Output:
A new probability density function *A'(k)*.

Steps:
1. For each profile *k* computes the following values:
   - P_R^k the probability of having followed the *R* path through arcs belonging to the profile *k*;
   - $\tilde{P}_{R_1,R_r}^k$ the reachability of the next node R_r starting from the first node R_1, through arcs belonging to the profile *k*;
   - *D[k]* the distribution of the visited nodes in *R* with respect to the profile *k*.
2. Computes the new value of *A'(k)* as follows:

   Increases *A(k)* for those *k* which have high P_R^k and vice versa;

   Increases *A(k)* for those *k* which have high $\tilde{P}_{R_1,R_r}^k$ and vice versa;

   Weights *A(k)* values with using the *D[k]* distribution.

To avoid an "infinite memory" effect, only the most recently followed *r-1* links (*r* nodes) are considered. As an example, if *R* was the path followed since the initial node, the probability P_R^k of having followed *R* in the profile *k* will be zero if the user has visited just one node not belonging to the profile *k*. Note that we consider $W_k(i,j) = 0$, if $(i, j) \notin E_k$, $k=1,\ldots M$.

The P_R^k takes in account the relevance of the followed path with respect to each profile.

The reachability of the next node starting from the first node in *R* takes in account the way the user could reach that node. In fact, a high reachability of R_r in the profile *k* means the user would have reached the next node in a more "natural" way by following the links of the profile *k*.

Temporary deviations that do not move the user's interests are taken in account trading off the effects of P_R^k and $\tilde{P}_{R_1,R_r}^k$ on *A(k)*. The former takes in account the actual path so aims to move towards the profile corresponding to recent preferences, whereas the latter aims to disregard recent (local) choices, as the "shortest" paths not necessarily consider the visited nodes between R_1 and R_r.

Moreover, the algorithm takes in account the distribution *D* of the visited nodes in *R* with respect to each profile, increasing probabilities of those profiles having high *D* values (i.e. profiles which comprise many nodes of *R*). If many of the recent nodes belong to a profile it is likely that the user belongs to that profile too.

Another factor that can be taken in account is the reading time of a page: the contribution of an arc can be weighted with the time the user spends on the page reached by that arc, with a threshold value.

7. System architecture

A system supporting the Probabilistic Application Domain and User models has been designed and partially implemented. The system has a three-tier architecture comprising the *User, Application* and *Data layers. User layer* corresponds to the browser. It carries out "light" computation because it receives HTML pages and executes only an invisible applet that signals to the server that the user is not interested to that page at the moment (e.g. has reduced to icon its window).

7.1. Application layer

At the *Application layer* (figure 2) there are three main modules: *Session Manager* (SM), *User Modelling Component* (UMC) and *Personalization Component* (PC) [3]; they run together with a Web-Server.

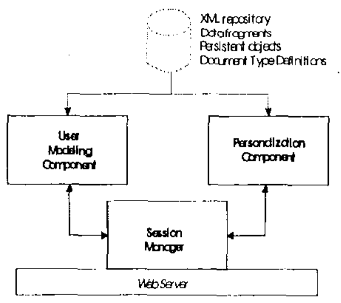

Figure 2. Application layer.

Session Manager responds to the user's actions and communicates them to the User Modelling Component. The UMC, which maintains the most recent actions of the user, executes the algorithm for the evaluation of belonging probabilities, evaluates the new profile, and gives the result back to the SM.

With the new profile the SM asks to the Personalization Component for the presentation unit, instantiated with respect to that profile and context variables. The PC dynamically generates the HTML code of the page: it extracts the XML document from the database and selects the fragments to compose the page, using the profile as a parameter. After having received the definitive page the SM sends it to the user.

In the definitive page the most likely links are generally more visible (e.g. showing a "next page" button or using different colours). Furthermore, it can be useful to cache the most recently visited pages, thus reducing the overhead of reconstructing recent HTML pages if profile does not change.

7.2. Data layer and Author Module

The main goal of the *Data layer* (figure 3) is to store persistent data and to offer efficient access primitives. It provides data to the Application Layer, allowing access to the Logical Navigation Graphs, XML documents and data fragments, respectively to the UMC and to the PC.

The Data layer includes:

- an object/relational multimedia database that stores basic data fragments and the DTDs,
- an XML repository that stores the XML documents,
- an object repository that stores persistent representations of graphs,
- and an *Author Module* used for the hypermedia design.

The *Author Module* features the following facilities:

- The *Graph authoring* utility generates persistent objects (e.g. Java Entity Beans) containing the weighted graphs, starting from a description of them contained in a text file or from a representation built using an internal GUI. The utility emphasises the existence of some high-probability cycles and asks for confirmation. The persistent objects contain all the data useful for the system as the shortest path for each pair of nodes.
- The *XML Editor* utility allows the editing of XML documents (graphs' nodes).
- The *XML Enterer* utility performs a validating parsing of XML documents with respect to the DTDs, verifies their congruence with graphs and stores them into the XML repository.

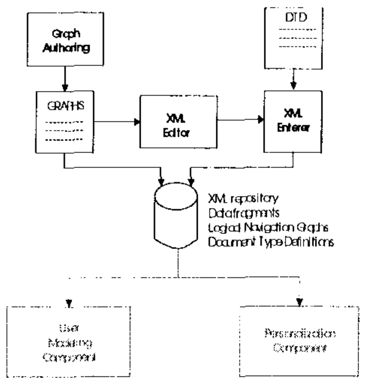

Figure 3. Data layer and Author Module.

8. Conclusions and future work

In this work we presented an architecture for the development of web-based Adaptive Hypermedia Systems based on stereotype profiles. The architecture uses graphs of XML documents to describe the application domain and a probabilistic model to adapt the web-site contents and links to the user's behaviour. The use of XML makes it possible to elegantly describe application domain, data access and dynamic data composition functions, allowing to separate the application data in different levels.

The user's behaviour is modelled using a probabilistic model: the main contribution of the paper is to combine information regarding the followed path, the "shortest" path (i.e. the path not necessarily followed but with higher probabilities) and a classification of the visited nodes, to calculate the distribution of the belonging probabilities (i.e. to estimate the profile that best fits the user). So, the most promising profile, that is a "view" over the application domain, is dynamically assigned to the user using that probability distribution. The views over the Application domain are currently static (with a fixed set of profiles) but it is possible to dynamically change the weights of the graphs' arcs on the basis of context change and user's behaviour.

We have implemented a preliminary prototype of the system in the domain of Cultural Assets using three profiles to model the user's behaviour: *generic*, *tourist* and *expert* (archaeologist). We used Java and relational databases to enhance portability.

Current and future work will concern the validation and tuning of the probabilistic model, the development of tools to improve and simplify the authoring phase and the support of dynamic content generation.

Acknowledgements

We wish to thank the anonymous referees for the useful comments and Domenico Saccà for his help on an early version of the manuscript.

References

[1] S. Abiteboul, B. Amann, S. Cluet, A. Eyal, L. Mignet, T. Milo, *Active views for electronic commerce*, in *Proceedings of the 25th VLDB Conference*, 1999.

[2] L. Ardissono, L. Console, I. Torre, *Exploiting user models for personalizing news presentations*, in *Proceedings of the 2nd workshop on adaptive systems and user modeling on the WWW*, 1998.

[3] L. Ardissono, A. Goy, R. Meo, G. Petrone, L. Console, L. Lesmo, C. Simone, P. Torasso, *A configurable system for the*

construction of adaptive virtual stores, World Wide Web Journal, Baltzer Science Publisher, 1999.

[4] P. Atzeni, G. Mecca, P. Merialdo, *To weave the Web*, in *Proceedings of the 23rd VLDB*, 1997.

[5] C. Boyle, A.O. Encarnacion, *Metadoc: An Adaptive Hypertext Reading System*, in *User Modeling and User-Adapted Interaction* 4(1): 1-19, 1994]

[6] P.Brusilovsky, *Methods and techniques of adaptive hypermedia*, in *User modeling and user adapted interaction*, v.6, n.2-3, 1996.

[7] P. Brusilovsky, *Efficient techniques for adaptive hypermedia*, in: C. Nicholas and J. Mayfield (eds.): *Intelligent hypertext: Advanced techniques for the World Wide Web.* Lecture Notes in Computer Science, Vol. 1326, Berlin: Springer-Verlag, pp. 12-30, 1997.

[8] S. Ceri, P. Fraternali, S. Paraboschi, *Data-driven one-to-one web-site generation for data-intensive applications*, in *Proceedings of the 25th VLDB Conference*, 1999.

[9] P. De Bra, *Design issues in adaptive web-site development*, in *Proceedings of the second workshop on adaptive systems and user modeling on the WWW*, 1999.

[10] P. De Bra, L. Calvi, AHA: *A generic adaptive hypermedia system*, in *Proceedings of the 2nd workshop on adaptive hypertext and hypermedia*, Pittsburgh, 1998.

[11] P. De Bra, P. Brusilovsky, G.J. Houben, *Adaptive Hypermedia: from systems to framework*, ACM Computing Surveys, Symposium Issue on Hypertext and Hypermedia, 1999

[12] J. Eklund, P. Brusilovsky, *The value of adaptivity in hypermedia learning environments: a short review fo empirical evidence*, in P. Brusilovsky and P.

[13] M. F. Fernandez, D. Florescu, A. Y. Levy, D. Sucin, *Catching the boat with Strudel: experiences with a web-site management system*, in *Proceedings of SIGMOD'98*, 1998.

[14] J. Fink, A. Kobsa, A. Nill, *User-oriented adaptivity and adaptability in the AVANTI project*, in *Designing for the Web: empirical studies*, Microsoft usability group, Redmond, 1996.

[15] J. Hothi, W. Hall, *An evaluation fo adapted hypermedia techniques using static user modelling*, in *Proceedings of the 2nd workshop on adaptive hypertext and hypermedia*, Pittsburgh, 1998.

[16] G.J. Houben, P.Lemmens, *A Software architecture for generating hypermedia applications for ad-hoc database output*, Technical Report Eindhoven University of Technology Department of Computing Science http://wwwis.win.tue.nl/~houben/pub/csr99-16.ps.

[17] HNC software Inc., *Aptex SelectCast*, http://www.aptex.com/Products/selectcast.

[18] J.Kay, *Vive la difference! Individualized interactions with users*, in *Proceedings of the 14th IJCAI*, Montreal, 1995.

[19] P. Maes, *Software agents: humanizing the global computer*, in *IEEE Internet Computing*, v.1, n.4, Luglio/Agosto 1997.

[20] Megginson Technologies, SAX 1.0: The simple API for XML, 1999.

[21] M. Perkowitz, O. Etzioni, *Adaptive sites: Automatically learning from user access patterns*, Technical Report, Department of computer science and engineering, University of Washington, 1997.

[22] M. Perkowitz, O. Etzioni, *Towards adaptive web sites: conceptual framework and case study*, Technical Report, Department of computer science and engineering, University of Washington, 1999.

[23] N.Walsh, *What do XML documents look like?*, XML.com Tutorial, 1998.

[24] World Wide Web Consortium, *Cascading Style Sheets Level 1*, Recommendation, 1996.

[25] World Wide Web Consortium, *Document Object Model Level 2 Specification*, Candidate Recommendation, 1999.

[26] [World Wide Web Consortium, *Extensible Markup Language*, Recommendation, 1998.

[27] World Wide Web Consortium, *Extensible Stylesheet Language*, Working draft, 2000.

[28] H.Wu, G.J. Houben, P. De Bra, *Authoring support for adaptive hypermedia applications*, Ed-Media 99 Conference.