Modeling Adaptive Hypermedia with an Object-Oriented Approach and XML

Mario Cannataro[1], Alfredo Cuzzocrea[1,2] Carlo Mastroianni[1], Riccardo Ortale[1,2], and Andrea Pugliese[1,2]

[1] ISI-CNR, Via P.Bucci, Rende, Italy
[2] DEIS-Università della Calabria

Abstract. This work presents an Application Domain model for Adaptive Hypermedia Systems and an architecture for its support. For the description of the high-level structure of the application domain we propose an object-oriented model based on the class diagrams of the Unified Modeling Language, extended with *(i)* a graph-based formalism for capturing navigational properties of the hypermedia and *(ii)* a logic-based formalism for expressing further semantic properties of the domain. The model makes use of XML for the description of metadata about basic information fragments and "neutral" pages to be adapted. Moreover, we propose a three-dimensional approach to model different aspects of the adaptation model, based on different user's characteristics: an adaptive hypermedia is modeled with respect to such dimensions, and a view over it corresponds to each potential position of the user in the "adaptation space". In particular, a rule-based method is used to determine the generation and deliver process that best fits technological constraints.

1 Introduction

In hypertext-based multimedia systems, the personalization of presentations and contents (i.e. their adaptation to user's requirements and goals) is becoming a major requirement. Application fields where content personalization is useful are manifold; they comprise on-line advertising, direct web-marketing, electronic commerce, on-line learning and teaching, etc. The need for adaptation arises from different aspects of the interaction between users and hypermedia systems. User classes to be dealt with are increasingly heterogeneous due to different interests and goals, world-wide deployment of information and services, etc. Furthermore, nowadays hypermedia systems must be made accessible from different user's terminals, through different kinds of networks and so on.

To face some of these problems, in the last years the concepts of user model-based adaptive systems and hypermedia user interfaces have come together in the *Adaptive Hypermedia* (*AH*) research theme [1, 5, 4]. The basic components of adaptive hypermedia systems are *(i)* the *Application Domain Model*, used to describe the hypermedia basic contents and their organization to depict more abstract concepts, *(ii)* the *User Model*, which attempts to describe some user's characteristics and his/her expectations in the browsing of hypermedia, and *(iii)* the *Adaptation Model* that describes how to adapt contents, i.e. how to the manipulate basic *information fragments* and the links. More recently, the capability to deliver a certain content to different kind of terminals, i.e. the support of multi-channel accessible web systems, is becoming an important requirement. To efficiently allow the realization of user-adaptable content and presentation, a modular and scalable approach to describe and support the adaptation process must be adopted. A number of interesting models, architectures and methodologies have been developed in the last years for describing and supporting (adaptive) hypermedia systems [11, 10, 14, 2, 8, 13, 12, 9].

In this paper we present a model for Adaptive Hypermedia and a flexible architecture for its support. Our work is specifically concerned with a complete and flexible data-centric support of adaptation; the proposed model has in fact a pervasive orientation towards such issues, typically not crucial in hypermedia models. We are intended to focus on *(i)* the description of the structure and contents of an Adaptive Hypermedia in such a way it is possible to point out the components

on which to perform adaptation (the *what*), and our belief is that the most promising approach in modeling the application domain is *data-centric* (in fact many recent researches employ well-known database modeling techniques); *(ii)* a simple representation of the logic of the adaptation process, distinguishing between adaptation driven by user needs and adaptation driven by technological constraints (the *how*); *(iii)* the support of a wide range of adaptation sources.

The logical structure and contents of an adaptive hypermedia are described along two different layers. The lower layer aim is to define the content of XML pages and the associated semantics (using an object-oriented model) and navigational features of the hypermedia (with a directed graph model). The upper layer describes the structure of the hypermedia as a set of views associated to *groups* of users (i.e. *stereotype profiles*) and some semantic relationships among profiles (using logical rules). Finally, the adaptation model is based on a multidimensional approach: each part of the hypermedia is described along three different *adaptivity dimensions*, each related to a different aspect of user's characteristics (behavior, used technology and external environment). A view over the Application Domain corresponds to each possible position of the user in the *adaptation space*. The XML pages are independent from such position, and the final pages (e.g. HTML, WML, etc.) to be delivered are obtained through a transformation that is carried out in two distinct phases, the first one driven by the user's profile and environmental conditions, and the second one driven by technological aspects.

2 Adaptive Hypermedia Modeling

In our approach to the modeling of adaptive hypermedia we chose to adopt the classical Object-Oriented paradigm, since it permits a complete high-level description of concepts. Furthermore, we chose to adopt XML as the basic formalism due to its flexibility and data-centric orientation. In fact, XML makes it possible to elegantly describe data access and dynamic data composition functions, allowing the use of pre-existing multimedia basic data (e.g. stored in relational databases and/or file systems) and the description of contents in a completely terminal-independent way.

We model the (heterogeneous) data sources by means of XML metadescriptions (Sec. 2.2). The basic information fragments are extracted from data sources and used to compose descriptions of pages which are "neutral" with respect to the user's characteristics and preferences; such pages are called *Presentation Descriptions* (*PD*). The PDs are organized in a directed graph for navigational aspects, and in an object-oriented structure for semantic purposes in the *Description Layer* (Sec. 2.3), while knowledge-related concepts (*topics*) are associated to PDs and profiles in the *Logical Layer* (Sec. 2.4). The transformation from the PDs to the delivered final pages is carried out on the basis of the position of the user in an *adaptation space* (Sec. 2.1). The process is performed in two phases: in the first phase the PD is instantiated with respect to the environmental and user dimension and a "technological independent" PD is generated. In the second phase (Sec. 2.5) the PD is instantiated with respect to the technology dimension.

2.1 Adaptation Space

In our proposal the application domain is modeled along three abstract orthogonal adaptivity dimensions:

- User's behavior (browsing activity, preferences, etc.);
- External environment (time-spatial location, language, socio-political issues, etc.);
- Technology (user's terminal, client/server processing power, kind of network, etc).

The position of the user in the *adaptation space* can be denoted by a tuple of the form $[B, E, T]$. The B value captures the user's profile; the E and T values respectively identify environmental

location and used technologies. The AHS monitors the different possible parameters that can affect the position of the user in the adaptation space, collecting a set of values, called *User*, *Technological* and *External Variables*. On the basis of such variables, the system identifies the position of the user.

The user's behavior and external environment dimensions mainly drive the generation of pages content and links. Instead, the technology dimension mainly drives the adaptation of page layout and the page generation process. For example, an e-commerce web site could show a class of products that fits the user's needs (deducted from his/her behavior), applying a time-dependent price (e.g. night or day), formatting data with respect to the user terminal and sizing data with respect to network bandwidth.

2.2 XML Metadata about Basic Information Fragments

Information fragments are the atomic elements used to build hypermedia contents; fragments are extracted from data sources that, in the proposed model, are described by XML meta-descriptions. The use of metadata is a key aspect for the support of multidimensional adaptation; for example, an image could be represented using different detail levels, formats or points of view (shots), whereas a text could be organized as a hierarchy of fragments or written in different languages. Each fragment is associated to a different portion of the multidimensional adaptation space. By means of meta-descriptions, data fragments of the same kind can be treated in an integrated way, regardless of their actual sources; in the construction of pages the author refers to metadata, thus avoiding too low-level access to fragments.

A number of XML meta-descriptions have been designed making use of *XML Schemas* [16]. They comprise descriptions of text (hierarchically organized), object-relational database tables, queries versus object-relational data, queries versus XML data (expressed in *XQuery* [15]), video sequences, images, XML documents and HTML documents. As an example, consider the following meta-description of a query versus XML data, expressed in XQuery:

```
<xquery alias="authorsquery">
  <statement>
    <![CDATA[
    <authors>
        {for $b in document("...")/bib/book
         where $b/subject = #key
         return
            <author>
            {$b/author/name}
            {$b/author/age}
            </author>
        }
    </authors>
    ]]>
  </statement>
  <result-structure>
    <value path-expression="/authors/author/name" alias="name"/>
    <value path-expression="/authors/author/age" alias="age"/>
    ...
  </result-structure>
</xquery>
```

In such meta-description the key elements are the statement (eventually with some parameters, `#key` in the example) and a description of the resulting XML document. Specific parts of the documents extracted by means of the above-shown query, could be described as follows:

```
<xdoc alias="book-author">
  <instance alias="db" description="database authors"
    location-type="xquery"
    location="authorsquery(#key=databases).name"
    schema="..."/>
  <instance alias="comp" description="compression author"
    location-type="xquery"
    location="authorsquery(#key=compression).name"
    schema="..."/>
</xdoc>
```

We emphasize that the meta-description of an XML document is abstracted with respect to its actual source, referred by means of location attributes; furthermore, many instances of the same document can be differentiated within the same meta-description (in the example above, authors of books having different subjects). *Complex* meta-descriptions allow direct referencing of single information fragments by means of aliases, dot-notation and parameters. For example, `book-author.db` in the previous example refers to the `db` instance of the `book-author` fragment, where `db` is in turn an alias for the query `authorsquery(#key=database).name`.

2.3 The Description Layer and the Presentation Descriptions

In the description layer the application domain is modelled as a directed graph, where nodes are the presentation descriptions and arcs represent links. Furthermore, we apply the object-oriented paradigm to capture the semantic relationships among the PDs: we define the PDs as objects which are instances of predefined classes.

Presentation descriptions are composed of four sections:

- The `OOStructureInfo` section includes information concerning the object-oriented organization of the domain. The interface of each class of PDs is composed of the set of ingoing and outgoing links. Furthermore, a *type* is associated to each link to express its semantics, and to define the *compatibility* among outgoing and ingoing links of different classes (e.g. a PD A can be linked to another PD B if the PD A has an outgoing link whose type is the same of the type of an ingoing link of the PD B). With regard to inheritance, a subclass inherits the information fragments and the links of the parent classes. The `OOStructureInfo` section is edited by means of a tool based on the *Unified Modeling Language* [17] (specifically on the class diagram), that allows the author to design the overall PD hierarchy of the application domain. Inside each PD, the tool automatically fills in the `OOStructureInfo` section, writing the O-O relationships among the PDs.
- The `AdapDimensionsInfo` section contains information about the instantiation of PDs with respect to the three adaptivity dimensions; this information describes how to extract fragments on the basis of the user position in the adaptation space, and which XSL stylesheet to apply in order to transform the PDs into the final pages to be delivered to the client. The `AdapDimensionsInfo` section is generated by means of a simple XML editor, on the basis of the author's domain knowledge.
- The `ContentLayout` section contains references to the information fragments to be used to compose the PDs.
- The `AuxInfo` section contains auxiliary information (e.g. data islands), which are not to be processed by the system itself, but can be used by the client (for example, they can contain embedded code for client applications), or by network lower layers.

In the following, we show the XML Schema structure of the `OOStructureInfo` section. It includes the name of the PD class (`entityName` element), a set of parent classes (`superEntityName`

elements) and the set of links that define the PD interface (`link` elements, each with an associated name and type).

```
<xs:element name = "entityName" type = "xs:string"/>
<xs:element name = "superEntityName" type = "xs:string"/>
<xs:element name = "linkName" type = "xs:string"/>
<xs:element name = "linkType" type = "xs:string"/>
<xs:attribute name = "direction" use = "required">
  <xs:simpleType>
    <xs:restriction base = "xs:string">
      <xs:enumeration value = "in"/>
      <xs:enumeration value = "out"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
<xs:element name = "link">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref = "linkName"/>
      <xs:element ref = "linkType"/>
    </xs:sequence>
    <xs:attribute ref = "direction"/>
  </xs:complexType>
</xs:element>
<xs:element name = "OOStructureInfo">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref = "entityName"/>
      <xs:element ref = "superEntityName" minOccours = "0" maxOccours = "unbounded"/>
      <xs:element ref = "link" minOccours = "1" maxOccours = "unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

2.4 The Logical Layer

The logical layer aim is to model the domain adaptivity with respect to stereotype user profiles. The logical model is a set of *Profile Views* (PV), where each PV is a view of the overall hyperspace domain associated to a user profile. The PV associated to the profile p includes all the PDs accessible by the users belonging to profile p, therefore it represents the hypermedia domain and the navigational space for that profile.

Following the two-layer definition of the application domain model, the model design is composed of two design phases, operating at two different abstraction levels. The first phase is related to the definition of the navigational directed graph: the author designs the presentation descriptions and their relations as hyperlinks. In the second phase the author identifies the user profiles and, on the basis of his/her domain knowledge, designs the PVs associated to each profile. The PV design can be carried out incrementally: for each PD, the author determines all the user profiles that can access it. At the end of the process, each PV is composed of all the PDs that have been associated to the corresponding user profile.

Each PV (and consequently each associated profile) corresponds to a set of topics that represents the knowledge contained in the PV (*knowledge description*). More precisely, this correspondence can be formally defined through a function $\kappa(\cdot)$, that can be applied to a single PD, or to a user profile, and returns the corresponding set of topics:

– $\kappa(\cdot)$, applied to the presentation description PD_i, returns the set of topics captured by PD_i;

– $\kappa(\cdot)$, applied to the user profile p, returns the knowledge domain of p: $\kappa(p) = \cup_i \kappa(PD_i) \mid PD_i \in PV_p$, where PV_p is the profile view associated to p.

The instantiation of a particular PD with respect to a given profile B and a given position along the external environment dimension E can be seen as the application of a function δ that transforms a given PD into a technology-independent XML page, called $\widehat{PD}$, which will be given as input to the multichannel layer (Sec. 2.5):

$$\delta : (PD, B, E) \longrightarrow \widehat{PD}.$$

Furthermore, we improve the logical description of the hypermedia structure by means of a *Semantic Precedence Operator*, $\leftarrow$, which is used to define constrains about profile changes. If applied to two knowledge domains, e.g. $\kappa(p_2) \leftarrow \kappa(p_1)$, the operator points out that a user cannot access topics related to the profile p_2 until he/she has accessed some topics included in the knowledge domain of p_1 (i.e. until he accesses the PDs that capture those topics). This constraint can be better specified by means of a *semantic precedence matrix*. This matrix has as many rows as the topics of $\kappa(p_2)$ and as many columns as the topics of $\kappa(p_1)$. Each element (i, j) of the matrix can assume a boolean value; *true* means that the user must visit a PD containing the topic j of the domain $\kappa(p_1)$ before his/her profile can change form p_1 to p_2 and the topic i of the domain $\kappa(p_2)$ can be accessed. So each row specifies which topic of $\kappa(p_1)$ a user belonging to the profile p_1 must know to change his profile to p_2 by entering a particular topic of $\kappa(p_2)$. For example, a semantic precedence matrix with all values equal to *true* means that, whatever is the entry point to the profile p_2, the user must have visited all the topics of $\kappa(p_1)$. An entry point of a profile p_2 is defined as a node (PD) that, when accessed through a hyperlink, allows the user to change his profile to p_2. The semantic precedence operator can be used in general logic rules; e.g. $\kappa(p_1) \leftarrow \kappa(p_2) \wedge (\kappa(p_3) \vee \kappa(p_4))$ is a rule that expresses a more complex relationship among the profiles involved.

2.5 Adaptation Model for the Technological Dimension

The technology dimension drives the adaptation of the page layout to the client device (PC, handheld computer, WAP device, etc.), and the page generation method. The technology adaptation is performed by means of a *Multichannel Layer*. In the following we will describe *(i)* the alternative page generation methods, *(ii)* the technological variables, and *(iii)* the multichannel layer.

Page generation methods. The final pages displayed on the client device (written in HTML, WML, etc.) are dynamically generated by performing a transformation of the $\widehat{PD}$ using a XSL document/program. Several XSL stylesheets can be used to transform the $\widehat{PD}$s, depending on the client device features. Moreover, three different page generation methods have been designed:

a. The page generation takes place entirely on the server. It picks out the Information Fragments, applies the transformation using the appropriate XSL document, and then sends the page (HTML, WML, etc.) to the client. The main drawback of this method is that the client cannot access the XML content. As an example, if the client is an application, e.g. a workflow or a distributed computing application, it could need to access and process the XML data.
b. Similar to the method *(a)*, but the server sends to the client HTML (WML) pages that contain XML data islands. These data are not processed by the server XSL processor, and are not displayed on the client device, but can be accessed by client programs.
c. The page generation is performed entirely on the client: the server sends to the client both the XML document and the XSL document that the client device will use to carry out the transformation.

The technological variables. The technological variables are used by the multichannel layer to adapt the presentation and the generation process to the client device. On our system we use five groups of technological variables:

1. Variables related to the XML and XSL support on the client device;
2. Variables addressing the client device processing power (client-side XSL formatting may take place only if the device can manage such a complex and time-consuming operation);
3. Variables related to the client device display features (resolution, dimensions, etc.);
4. Variables concerning the kind of client data usage (e.g., it is useful to know whether or not clients need to access and process the "pure" XML data);
5. Variables related to the server processing workload.

The variables belonging to the first three groups are extracted from the *device knowledge base* (shown in the next Section), while data usage variables are associated to the client (e.g. they can be associated to the user profile), and group 5 variables are determined by the server.

The Multichannel Layer. The main components of the multichannel layer are the *Device Knowledge Base* (*DKB*) and the *Presentation Rules Executor* (*PRE*).

The device knowledge base is a repository composed of a set of entries, each describing the features of a specific client device. For each client request, the client device is determined according to the data contained in the *User Agent* field of the request, and the corresponding device entry is selected. Each entry is composed of variable-value pairs, where variables correspond to the technology variables belonging to the first three groups.

The presentation rules executor determines the presentation layout and the page generation method based on a set of rules that check the values assumed by the technological variables. Rules are defined in an ad hoc XML syntax, and are modelled according to the well known *event-condition-action* (*ECA*) paradigm. The following XML fragment shows a typical presentation rule.

```
<rule id="1">
    <conditions>
        <technological-variable group="1" xml-support="yes"/>
        <technological-variable group="2" processing-power="high"/>
        <technological-variable group="3" display-area="small" res-value="medium"/>
        <technological-variable group="4" xmldata-need="no"/>
    </conditions>
    <action>
    <xsl-formatting value="clientside" method="c" stylesheet="AdHocStylesheet.xsl"/>
    </action>
</rule>
```

The rule states that if the client does not need to elaborate the XML data content (`xmldata-need` belonging to group 4), the client device fully supports XSL transformation (group 1 and 2 variables), and display features are appropriate for the data to visualize (group 3 variables), the PRE chooses the page generation method *c* (client-side XSL formatting), and the XSL stylesheet `AdHocStylesheet.xsl`.

Note that the above rule does not consider the server-side workload. Another presentation rule may state that XSL formatting has to take place on the server side (even if the client device supports XML and XSL) if the client does not need to access XML data, unless the server workload exceeds a specified workload threshold.

Events of the ECA paradigm are implicitly managed by the adaptive system and correspond to the user choice of a given PD. Conditions correspond to checks on the technological variable

values. Actions are performed when a logical expression (a *presentation rule*), composed of atomic conditions, is evaluated. The above XML syntax can be used to compose complex presentation rules, based on all the possible combinations of technological variable values. Actions mainly consist of the choice of the page generation method and the choice of the most suitable XSL stylesheet that will drive the XSL formatting.

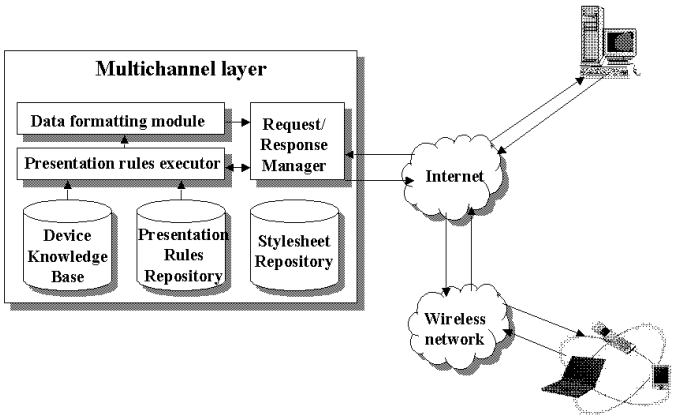

Fig. 1. The multichannel layer.

Figure 1 shows the multichannel layer architecture. This is a logical layer that uses components belonging to the three architectural tiers (see Section 3). The Request/Response Manager processes client requests (coming either from a wired or a from a wireless device), and after the generation of the $\widehat{PD}$s passes both to the PRE. The PRE extracts the user-agent data from the requests and accesses the device knowledge base to evaluate the technological variables. Then, the PRE executes the presentation rules contained in the presentation rules repository, according to the ECA paradigm.

The PRE chooses the page generation method and the XSL stylesheet (picking it out form the Stylesheet Repository). In the case of client-side formatting (page generation method c), the PRE passes both the XML data (the $\widehat{PD}$) and the selected stylesheet to the Request/Response manager, which in turn sends them to the client. In the case of server-side formatting (page generation methods a or b), the PRE activates the Data Formatting Module, which converts the XML data according to the XSL directives, and then passes the formatted data to the Request/Response Manager, which in turn sends it to the client.

3 An Architecture for the Support of the Proposed Model

We have designed and are currently implementing an architecture for the support of the proposed model, comprising an authoring suite and a run-time system.

The main tasks of the authoring suite (named *Java Adaptive Hypermedia Suite – JAHS*) are the following: *(i)* composition of the UML class diagrams; *(ii)* composition of the semantic precedence logic rules; *(iii)* transformation of the class diagrams and semantic precedence rules into an XML formalism; *(iv)* browsing of the data sources and composition of the XML metadescriptions; *(v)* construction of the presentation descriptions; *(vi)* specification of the event-condition-action rules for the instantiation of the PDs with respect to the technological dimension. For space reasons, we do not further detail the authoring suite; the interested reader can refer to [6].

The run-time system has a three-tier architecture (Fig. 2), comprising the *User*, the *Application* and the *Data* tiers. The user tier receives final pages to be presented and eventually scripts or applets to be executed (e.g. for detecting local time, location, available bandwidth). The user's terminal and the terminal software (operating system, browser, etc.) are communicated by the terminal *User Agent* (e.g. the browser).

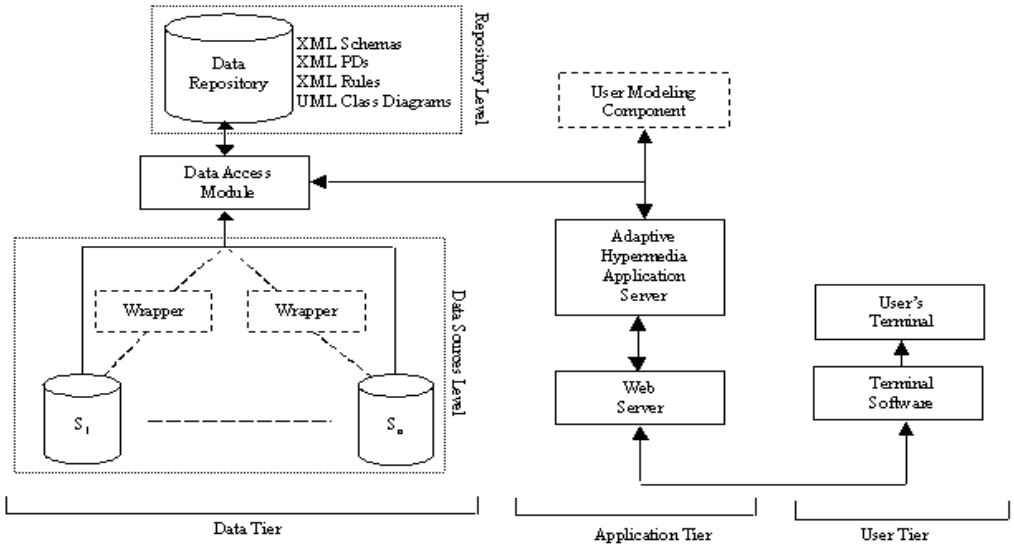

Fig. 2. The run-time system architecture.

The *User Modeling Component* (*UMC*) maintains the most recent actions of the user and evaluates them giving as a result the user's profile. In this paper we do not address such issue, demanding it to an external module; in our previous work we have proposed an approach in which hypermedia links are mapped into graph edges weighted with the probability of following them, and a probabilistic algorithm is used to estimate the user's profile ([7]).

The *Adaptive Hypermedia Application Server* (*AHAS*) runs together with a Web Server. It executes the following steps: *(i)* it communicates to the UMC the most recent choices of the user; *(ii)* it extracts from the XML repository the PD to be instantiated; *(iii)* it extracts the basic data fragments from the data sources on the basis of the user position; *(iv)* it interacts with the modules of the multichannel layer in order to generate the final page.

The data tier comprises the *Data Sources Level*, the *Repository Level* and a *Data Access Module*. The data sources level is an abstraction of the different kinds of data sources; each data source S_i is also accessed by a *Wrapper* software component, which generates the XML metadata describing the data fragments stored in S_i. The Repository Level is a common repository for data provided by the Data Source Level or produced by the author. It stores *(i)* XML documents including the Presentation Descriptions, metadata, Schemas, XSL stylesheets, and the active rules shown in Sections 2 and 2.5; *(ii)* the UML class diagrams representing the logical structure of the hypermedia. Finally, the data access module implements an abstract interface for accessing the data sources and the repository levels.

4 Concluding Remarks

In this paper we presented a model for adaptive hypermedia systems using the object-oriented paradigm and XML. An adaptive hypermedia is modeled considering a three-dimensional adaptation space, including the user's behavior, technology, and external environment dimensions. The adaptation process is performed evaluating the proper position of the user in the adaptation space, and transforming "neutral" XML pages according to that position.

We believe that the main contributions of this paper are:

- a new data-centric model to describe adaptive hypermedia specifically concerned with a flexible and effective support of the adaptation process; the model integrates a graph-based description of navigational properties and an object-oriented semantic description of the hypermedia, and uses a logical formalism to model knowledge-related aspects;
- a flexible and modular architecture for the run-time support of adaptive hypermedia systems, with particular regard to the adaptation with respect to technological aspects.

References

1. Adaptive Hypertext and Hypermedia Home Page, `http://wwwis.win.tue.nl/ah/`.
2. M. Bordegoni, G. Faconti, S. Feiner, M.T. Maybury, T. Rist, S. Ruggieri, P. Trahanias, and M. Wilson, "A Standard Reference Model for Intelligent Multimedia Presentation Systems", in *Computer Standards and Interfaces* 18, 1997.
3. P. Brusilovsky, "Methods and techniques of adaptive hypermedia", in *User Modeling and User Adapted Interaction*, v.6, n.2-3, 1996.
4. P. Brusilovsky, A. Kobsa, J. Vassileva (eds.), *Adaptive Hypertext and Hypermedia*, Kluwer Academic Publishers, 1998.
5. P. Brusilovsky, O. Stock, C. Strapparava (eds.), *Adaptive Hypermedia and Adaptive Web-Based Systems*, Proceedings of the International Conference AH 2000, Trento, Italy, 2000.
6. M. Cannataro, A. Cuzzocrea, A. Pugliese, "A multidimensional approach for modelling and supporting adaptive hypermedia systems", *Proceedings of the International Conference on Electronic Commerce and Web Technologies (Ec-Web 2001)*, Munich, Germany, 2001. LNCS 2115, pp. 132-141, Springer Verlag, 2001.
7. M. Cannataro, A. Cuzzocrea, A. Pugliese, "A probabilistic approach to model adaptive hypermedia systems", *International Workshop on Web Dynamics*, in conjunction with the *International Conference on Database Theory*, 2001.
8. S. Ceri, P. Fraternali, A. Bongio, "Web Modelling Language (WebML): a modelling language for designing web sites", *WWW9 Conference*, 2000.
9. P. De Bra, G.J. Houben, H. Wu, "AHAM - A dexter-based reference model for adaptive hypermedia", in *Proceedings of the ACM Conference on Hypertext and Hypermedia*, 1999.
10. F. Garzotto, D. Paolini, D. Schwabe, "HDM - A model-based approach to hypermedia application design", *ACM Transactions on Information Systems*, 1993.
11. F. Halasz, M. Schwartz, "The Dexter hypertext reference model", *CACM* v. 37, n. 2, 1994.
12. M. F. Fernandez, D. Florescu, A. Y. Levy, D. Suciu, "Catching the boat with Strudel: experiences with a web-site management system", in *Proceedings of SIGMOD'98*, 1998.
13. G. Mecca, P. Atzeni, A. Masci, P. Merialdo, G. Sindoni, "The Araneus Web-Based Management System", in *Exhibits Program of ACM SIGMOD '98*, 1998.
14. D. Schwabe, G. Rossi, "An object-oriented approach to Web-based applications design", in *Theory and Practice of Object Systems*, October 1998.
15. World Wide Web Consortium, The XML Query language, `http://www.w3.org/XML/Query`.
16. World Wide Web Consortium, The XML Schema definition language, `http://www.w3.org/XML/Schema`.
17. The Unified Modeling Language. `http://www.uml.org`.