A MODEL-BASED APPROACH TO WEB-APPLICATION DEVELOPMENT*

Oscar Diaz, Felipe Ibanez, Jon Iturrioz
The EKIN team
Dpto. de Lenguajes y Sistemas Informaticos
University of the Basque Country
Apdo. 649 - 20080 San Sebastin (Spain)
jipdigao@si.ehu.es, jibibanf@si.ehu.es, jipitsaj@si.ehu.es

Abstract The increasing growth in size and complexity of portals calls for a systematic way to web-application development that is able to face the stringent demands imposed on both the development and maintenance of these systems. Model-based approaches have been proposed to mitigate this situation. These approaches aim to find models, preferably orthogonal, that allow designers to declaratively specify a distinct concern of the application without being immediately immersed in details of implementations. This paper presents *AtariX*, a model-based tool that renders HTML pages from the declarative schemata specified by the designer. Each concern is described by a separate XML document: how data is integrated and structured (the content document), the topology of links (the navigation document) and the layout of each element (the presentation document). An application is then conformed by a set of schemata (i.e. model instances), for each of the distinct models. Separation of concerns and declarativeness enhance the application maintenability, as well as promoting concurrent development as each model can be assigned to a different team. *AtariX* has been fully implemented and its use is illustrated by designing and delivering a website for a scientific conference.

1. Introduction

Web-application development is currently suffering from a severe bottleneck as the gap between available implementation tools and application's requirements is enlarging. These difficulties are likely to become even more severe when web masters are trying to maintain these applications, particu-

*This research was partially supported by the Secretaría de Estado de Política Científica y Tecnológica of the Spanish Government under contract TIC 1999-1048-C02-02. Felipe Ibáñez enjoys a pre-doctoral grant by the University of the Basque Country.

larly, in the area of e-commerce [10]. In today´s e-commerce world, companies should adapt to changing conditions and the rapid evolution of the web-technology. However, it is a frustrating experience to see how often the web site bottleneck slows and restricts the evolution of the organization the site is supposedly serving.

In response to this need, distinct projects have been launched which aim at providing design guidelines and supporting tools for systematic web site construction [6]. One of the most frequently cited guidelines is splitting requirements into content concerns, navigation concerns and presentation concerns by using a model-based approach [9].This approach aims to find declarative models, preferably orthogonal, that allow designers to declaratively specify a distinct concern of the application without being immediately immersed in details of their implementation. An application is then conformed by a set of schemata (i.e. model instances) which describe distinct aspects of the application.

This paper presents *AtariX*, a model-based tool for web-application development that renders HTML pages from the declarative schemata specified by the designer. Each concern is described by a separate XML document: how data is integrated and described (the content document), the topology of links (the navigation document) and the layout of each element (the presentation document).

This paper shares objectives with efforts in the area of hypertexts. [8], [5] or [11] come from this area and their main focus is on providing powerful built-in navigation primitives. For data-intensive web sites (i.e. sites displaying primarily data that has been strored in a database) the *Torii* system stands out ([2], [3]). *Torii* is also a model-driven system which can be used as a powerful Web front-end for data stored on a database system. Unlike our approach, its content model is based on database views whereas *AtariX*'s content model is based on the XML data model.

The rest of the paper is structured as follows. The content model, the navigation model and the presentation model are presented in sections 2, 3 and 4, respectively. Finally, the conclusions are given. The IFIP conference web site is used as a running example throughout the paper . The implementation of this site can be found at *http://sipl68.si.ehu.es/atarix/ ifipExample.*

2. The content model

For our purposes, content has to do with two issue: its nature and its source. The former refers to whether data or documents are handled [1]. Data structure is characterized by being regular, fine-grained and the order is often not significant. In the conference web-site, attendee data follows this pattern: the same data must be collected for each attendee, the granularity of this data is

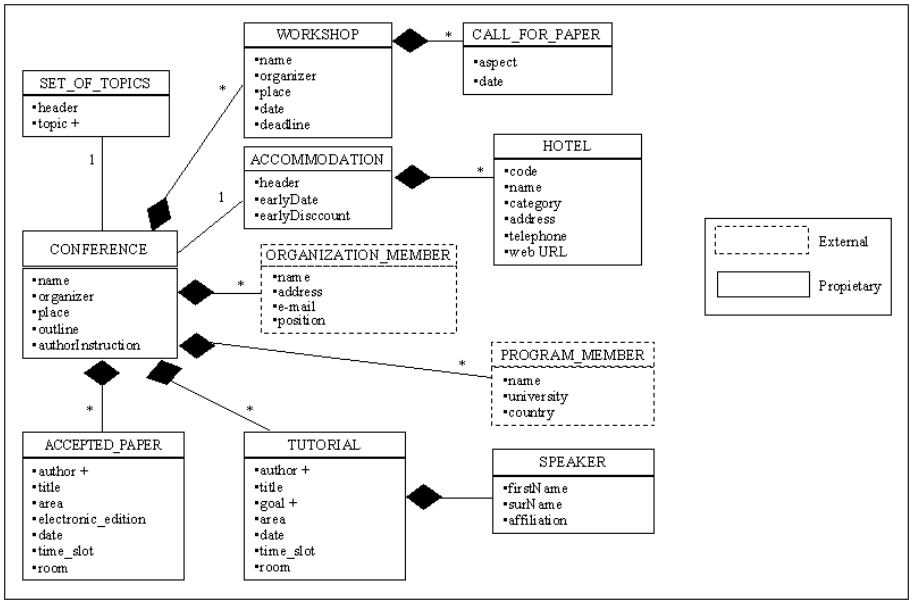

Figure 1. The content diagram for the conference example.

often atomic, and the order in which this data is arranged is not important. By contrast, document structure commonly follows an irregular structure, the data is larger-grained and the order is significant. For example, the introductory information of a conference, i.e. the outline, the topics of interest, the distinct deadlines to authors or the instructions to authors, have an irregular structure. In order to accommodate the idiosyncrasies of each conference, these aspects do not follow a common, regular structure. For instance, a specific conference can have a boat trip or provide strict guidelines for author's manuscripts, whereas other conferences do not include any of these aspects. Even for the very same notion (e.g. a banquet) different conferences impose distinct structures (e.g. is there a reception previous to the banquet? should a map be included? is dress-code required?). Variation is at the heart of the document notion.

The XML specification holds the potential for devising effective integration of both data and document-centric sources and is being widely adopted as a standard for information representation and exchange. The W3C is promoting distinct standards for document schema definition. This work uses XML-Schema [13]. An XML-Schema document is an XML document that describes the structure definition of a particular class of documents, i.e. that is addressing aspects such as which elements can occur and how they can be nested in the XML document that conforms to the schema.

```
<?xml version="1.0"?>
<?atarix:navigation_document href="conference.navigation.xml" type="text/xml"?>
<?atarix:presentation_document href="conference.presentation.xml" type="text/xml"?>
<?atarix:connection_document href="conference.DBConnection.xml" type="text/xml"?>
<?atarix:workview_document href="conference.workview.xml" type="text/xml"?>
<CONFERENCE xmlns:atarix="x-schema:../schemas/atarixSchema.xml">
  <NAME> IFIP 2.6 WORKING CONFERENCE ON DATABASE SEMANTICS (DS-9)</NAME>
  <ORGANIZER>International Federation for Information Processing </ORGANIZER>
  <WORKSHOP>
    <NAME>SEMANTIC ISSUES IN e-COMMERCE SYSTEMS</NAME>
    <ORGANIZER>IFIP Working Group 2.6 (Database)</ORGANIZER>
    <CALL_FOR_PAPER> ... </CALL_FOR_PAPER> ...
  </WORKSHOP>
  <SET_OF_TOPICS>  ...  </SET_OF_TOPICS>
(a) <atarix:QUERY rowName="ORGANIZATION_MEMBER" connection="xmldemo" >
      <atarix:SELECT> NAME, ADDRESS, EMAIL, POSITION</atarix:SELECT>
      <atarix:FROM>organizationMemberTable</atarix:FROM>
    </atarix:QUERY>
  <PROGRAM_MEMBER>...  </PROGRAM_MEMBER>
  <ACCEPTED_PAPER>
    <AUTHOR>Kenneth R. Jacobs</AUTHOR>
    <TITLE> Innovation in Database Management: Computer Science vs Engineering.</TITLE>
    <AREA>Data Management</AREA>
    <DATE>April 25, 2000</DATE>
    <ROOM>1</ROOM>
    <TIME_SLOT> 9:00 - 10:00 </TIME_SLOT>
  </ACCEPTED_PAPER>
  <TUTORIAL> ... </TUTORIAL> ...
  <ACCOMMODATION> ... </ACCOMMODATION>
</CONFERENCE>
```

Figure 2. The content document.

Although UML is not as expressive as XML-Schema (structure-wise), figure 1 depicts an UML diagram for the conference example. A conference content includes data about the aims of the conference, the organizing and program committee, the call-for-papers, the list of accepted papers and so on[1]. The XML-Schema counterpart is not shown here. However, how this schema is instatiated for an IFIP Conference can be found in figure 2.

As for the content sources, web-site development strives to unify distinct and potentially heterogeneous data sources. According to this aspect, content can be characterized as proprietary or external. Proprietary content belongs to the web-site and is provided within the content document itself. For instance, content about the aims of the conference, author instructions and etc. are examples of proprietary content; proprietary content is not shared with other sites. By contrast, external content is normally held on a database. A dotted line is used in figure 1 to indicate external content. For example, conference organization members data is retrieved from a database. This situation is indicated through the *QUERY* tag (see figure2a). The content of the *ORGA-*

[1]Although the UML notation is used, it is worth noticing that the rectangles do not stand for classes nor entities. They refer to sub-documents, better said, XML elements (i.e.: tags) of the content document. How these elements have been ascertained from use-cases or why these elements have been chosen to be explicitly represented in the UML diagram among the potentially large set of elements of the content document is out of the scope of this paper.

NIZATION_MEMBER element is the result of a database query. At run-time, the processing of the *QUERY* tag causes the attached query to retrieve the tuples from the *"memberTable"* table. This configuration states how many tuples should be retrieved each time depending on how many are needed. This query returns an XML document fragment with many child nodes named *ORGANIZATION_MEMBER(OM)* (the *rowName* attribute), and *NAME, ADDRESS* and *POSITION* as its sub-elements. It is worth pointing out that this tag is interpreted only when the user navigates to the *OM* element (see section 3). Stated another way, the query is executed only when OM data is required. Even though some data sources are likely to continue using relational database systems as a primary form of storage, we expect that most data sources will eventually provide XML access for their published data. A rudimentary mechanism is already in place for some database management systems such as Oracle.

The content document is situated at the center of the design lifecycle. All other aspects of the application are built up around this document. These other concerns are also described through XML documents, which will be discussed in more details in the next sections. In order to associate these documents with a given content document, the XML approach is followed. The processing instruction element *"xml:stylesheet"* attached to an XML document, indicates the XSLT document to be used for rendering an XML document. Likewise, *AtariX* provides distinct processing instruction elements (i.e. *atarix:navigation_document, atarix:presentation_document, atarix:workview_document* and *atarix: connection_document)* to indicate the navigation, presentation, workview and connection documents which describe how these concerns are realized by the associated content document (see figure 2).

3. The navigation model

As shown in the previous section, *AtariX* supports "the content space" by means of an XML document. The navigation model addresses how this document can be traversed through links. Figure 3 shows part of the navigation diagram for the conference example where the different arrows represent the possible links along "the content space". This diagram is realized through the navigation document (see figure 4). Each arrow of the navigation diagram maps to a *LINK* element in the navigation document. As an example, consider the link that leads from a */CONFERENCE/WORKSHOP* node to its related *CALL_FOR_PAPERs* nodes. This is expressed in *AtariX* as follows:

```
<LINK title="Call For Papers"
      from="/CONFERENCE/WORKSHOP" to="CALL_FOR_PAPER">
  ...
</LINK>
```

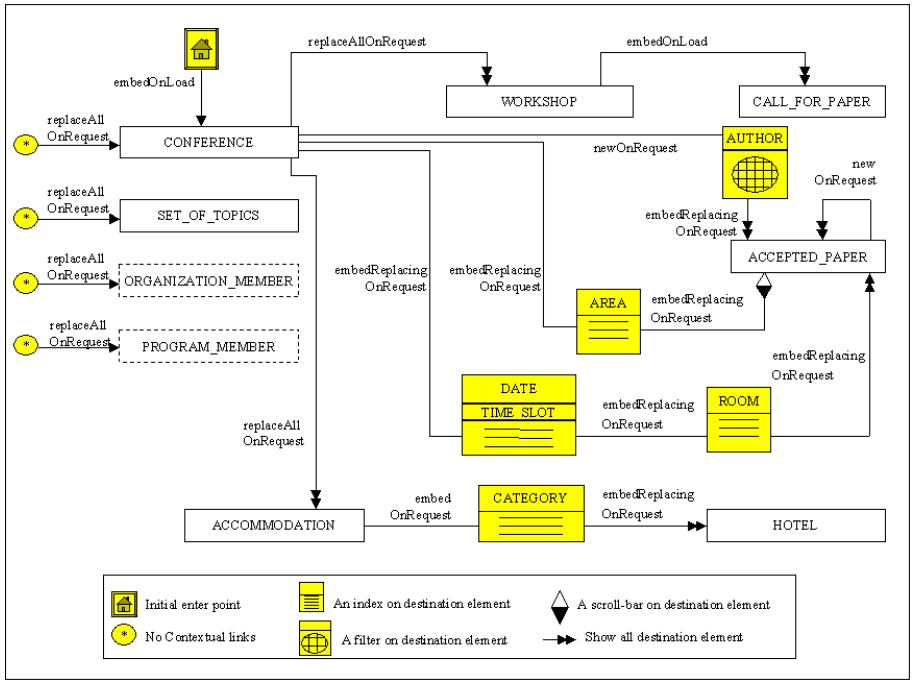

Figure 3. The navigation diagram for the conference example.

```
<atarix:LINKS xmlns="x-schema:../schemas/atarixSchema.xml">
  <!-home link-->
  <atarix:LINK title="Conference" from="home" to="/CONFERENCE">
    <atarix:CONTENT order="1" couplingMode="embedOnLoad"/>
  </atarix:LINK>
  <!-contextual links-->
  <atarix:LINK title="Workshops" from="/CONFERENCE" to="WORKSHOP">
    <atarix:CONTENT order="1" couplingMode="replaceAllOnRequest"/>
  </atarix:LINK>
  ...
<!-- No contextual links -->
  <atarix:LINK title="Conference" from="*" to="/CONFERENCE">
    <atarix:CONTENT order="1" couplingMode="replaceAllOnRequest"/>
  </atarix:LINK>
  <atarix:LINK title="Conference Organization" from="*" to="/CONFERENCE/ORGANIZING_MEMBER">
    <atarix:CONTENT order="1" couplingMode="replaceAllOnRequest"/>
  </atarix:LINK>
  ...
</atarix:LINKS>
```

Figure 4. The navigation document.

LINK is an XML element of the *AtariX* vocabulary. This element has a set of attributes which describe (1) the label of the link when rendered on the screen (the *title* attribute); (2) the origin of the link (the *from* attribute) that indicates when the link is available (in this case when the *WORKSHOP* element is rendered); and (3) the destination of the link (the *to* attribute) which states the element to be rendered when this path is followed. Besides the origin and destination, a *LINK* is characterized by a navigation and a coupling mode. Each of these aspects will be addressed in the next paragraphs.

Anchor identification. The distinct element tags of the XML content document are conceived as potential anchors from which to define origins and destinations. Elements within the XML content document are addressed using the W3C standard XPath notation[12]. XPath conceives an XML document as a tree of nodes, and follows a notation similar to the UNIX directory paths to address each node within the document (a *location path* using XML parlance). We have extended this notation to accommodate further navigation requirements.

Four kinds of location paths are distinguished: relative location paths, absolute location paths, the non-contextual location path, and the 'home' location path. The former two are standard XPath notations. Their main difference stems from what is the origin of the path: the *context node* (i.e. the current node being visited) for relative location paths, or the root of the document (denoted by "/") for absolute location paths. Relative and absolute location paths allow to address the descendents of a given node (either the current node or the root node), and therefore, support forward navigation. In the example shown previously, the *from* attribute holds an absolute path (i.e. */CONFERENCE/WORKSHOP*) whereas the *to* attribute is a relative path, i.e. it refers to the *CALL_FOR_PAPER* nodes which belong to a concrete *WORKSHOP* node.

However, backward navigation can also be useful, i.e. the location of the predecessors of the context node. To support backward navigation, the ".." notation has been introduced. For example, when trying to reach from an *ORGANIZATION_MEMBER* node, a sibling node (e.g. a *PROGRAM_MEMBER* node), the link destination path would be stated as *"../PROGRAM_MEMBER"*.

As for the non-contextual location path, it supports entry points to "the content space" regardless of the current position. (i.e. the current context or node being visited). This path is denoted by a "*" and can only be used as value for the link's origin. For instance, if the *from* attribute of the *LINK* element in the previous example had been set to "*", this would have permitted to reach the distinct *CALL_FOR_PAPER*s nodes from any point in the navigation space. In rendering terms, this means that any page will have the *"Call For Papers"* link. Finally, the home location path denotes the startup links. This path is indicated by *a "home"* value for the link's origin. If in the previous example

"home" had been the value of the *from* attribute, the content of the distinct *CALL_FOR_PAPER* nodes would have been rendered at the home page.

Figure 3 shows a possible navigation diagram for the conference example. As indicated by the *home* location path, the *CONFERENCE* outline is readily presented once connected to the site. Several non-contextual links are always available to display the *ORGANIZATION*_MEMBER, the *PROGRAM*_MEMBER or the *SET_OF_TOPICS* data within the conference scope. By contrast, *WORKSHOP* data can only be accessed while being on the conference outline, which in turn, leads the way to the *CALL_FOR_PAPER*s data. Each arc in this diagram is mapped to a *LINK* element in the navigation document (see figure 4).

As a final example, consider a recursive link that leads from an *ACCEPTED_PAPER* to the other accepted papers by the same author:

```
<LINK title="Other papers by the same author"
   from="/CONFERENCE/ACCEPTED_PAPER"
   to="/CONFERENCE/ACCEPTED_PAPER
      [AUTHOR=$FROM/AUTHOR] [TITLE !=$FROM/TITLE]">
  ...
</LINK>
```

Both, the *from* and *to* attributes hold an absolute path. However, the *to* path just addresses those *ACCEPTED_PAPER* elements which have the same *AUTHOR* as (and distinct *TITLE* from) the accepted paper that is being visited. At run time, the variable *$FROM* keeps the *ACCEPTED_PAPER* node which is being visited[2].

Navigation mode. Navigation proceeds from a node to those destination nodes of the chosen link. If a single destination node is available, navigation is straightforward. However, this is rarely the case, and in most cases navigation is one-to-many. The navigation mode indicates how to proceed in this case.

As an example, a *CONFERENCE* element can include a set of *ACCEPTED_PAPER* sub-elements. When traversing a link from a conference to its accepted papers, should all of them be processed at once? or is it preferable to browse them one by one? In this case, the designer is confronted with the decision of how the set of papers is traversed. This navigation mode is described by means of sub-elements of the *LINK* element. Based on the WebML modeling language [3], *AtariX* supports indexes, filters, and scrolls. A link can sequence some of these constructors to build up an aggregate navigation mode.

An index provides a shortcut to reach the desired destination nodes. For example, the way how papers are obtained through an index hierarchy is shown in figure 5: first, a *date+timeSlot* index is presented from where the user selects

[2]If the origin paper has more than one author, the link goes to those papers having at least one author in common.

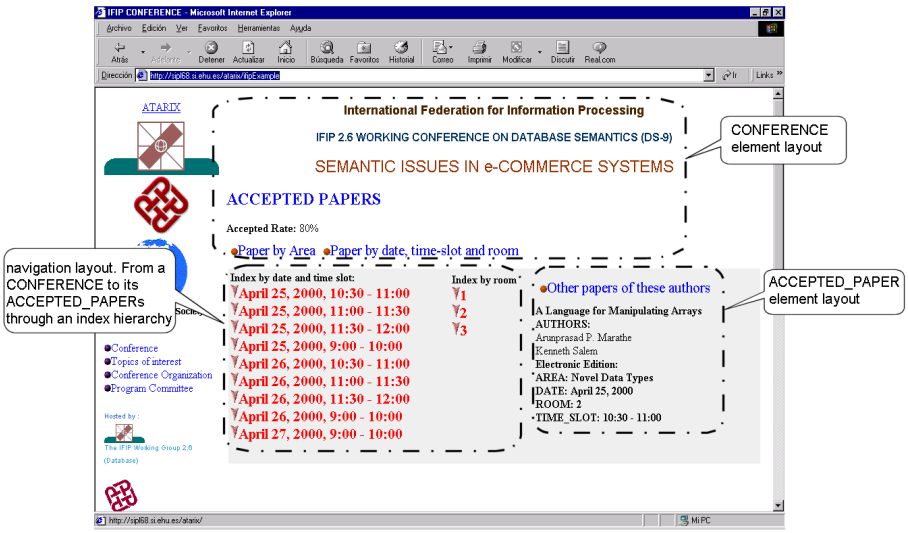

Figure 5. Navigation from *Conference* to a*cceptedPaper*: an index hierarchy along accepted papers.

a value; then, the system dynamically generates a second index of the rooms associated with the chosen *date+timeSlot* value; finally, once a *room* has been selected, the system retrieves the data of the paper to be presented in this room.

Index description is achieved through the *INDEX* element. For instance, the index hierarchy shown in figure 5 was obtained from the following description:

```
<LINK title="Papers by date,time-slot and room"
      from="/CONFERENCE" to="ACCEPTED_PAPER">
 <INDEX order="1" title="Index by date and time-slot"
        couplingMode="embedReplacingOnRequest">
   <INDEX_PROPERTY propertyName="DATE"/>
   <INDEX_PROPERTY propertyName="TIME_SLOT/>
 </INDEX>
 <INDEX order="2" title="Index by room" couplingMode="embedReplacingOnRequest">
   <INDEX_PROPERTY propertyName="ROOM"/>
 </INDEX>
 <CONTENT order="3" couplingMode="embedReplacingOnRequest">
</LINK>
```

This link specifies how is traversed the set of *ACCEPTED_PAPER* nodes (the *to* attribute) from a *CONFERENCE* node (the *from* attribute). The *INDEX_PROPERTY* sub-element indicates which property of the destination element (i.e. *ACCEPTED_PAPER*) serves as an index. By "property" is meant either an attribute or a sub-element. In the previous example, the system offers a first index based on *DATE* and *TIME_SLOT*, both are sub-elements of the *ACCEPTED_PAPER* element. As the number of papers is still large, for each

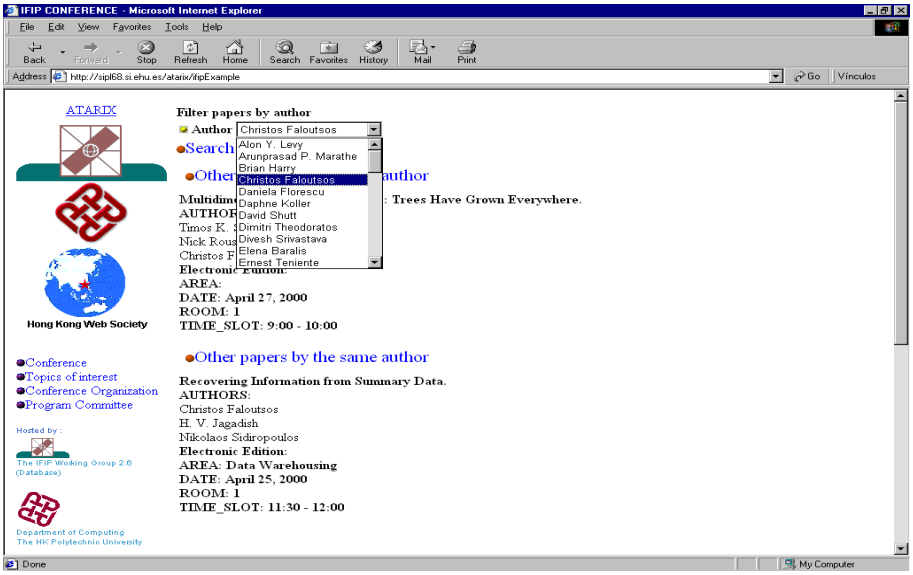

Figure 6. A filter is used to navigate from *Conference* to *acceptedPaper*. The user is asked to select an author from the range of possible authors.

date+timeSlot value, a second index is used to arrange papers by the *ROOM* in which they are presented. Notice that order of index elements is important.

If the set of destination nodes is large, a filter might be more suitable. A filter restricts the targeted nodes through a query. As an example, figure 6 shows a possible layout of a filter where only the papers that match the selected *AUTHOR* are rendered. The user is asked to select an author from the range of possible authors, and then, the whole set of papers by this author is presented at once. This navigation mode is specified in AtariX as follows:

```
<LINK title="Papers by author" from="/CONFERENCE" to="ACCEPTED_PAPER">
  <FILTER order="1" title="Filter papers by author" couplingMode="newOnRequest">
    <SEARCH_PROPERTY title="Author" propertyName="AUTHOR"
                     predicate="belongsTo"/>
  </FILTER>
  <CONTENT order="2" couplingMode="embedReplacingOnRequest"/>
</LINK>
```

The *FILTER* element encompasses the definition of a filter. The *SEARCH_PROPERTY* sub-element indicates an input field. This sub-element has three attributes which describe: (1) the label of the field when rendered on the screen (the *title* attribute); (2) the property of the destination element to search for (the *propertyName* attribute); and (3) the predicate to be satisfied by the destination nodes in order to be rendered (the *predicate* attribute). The *predicate* attribute can hold three values: *"belongsTo"*, which requires the introduced value to

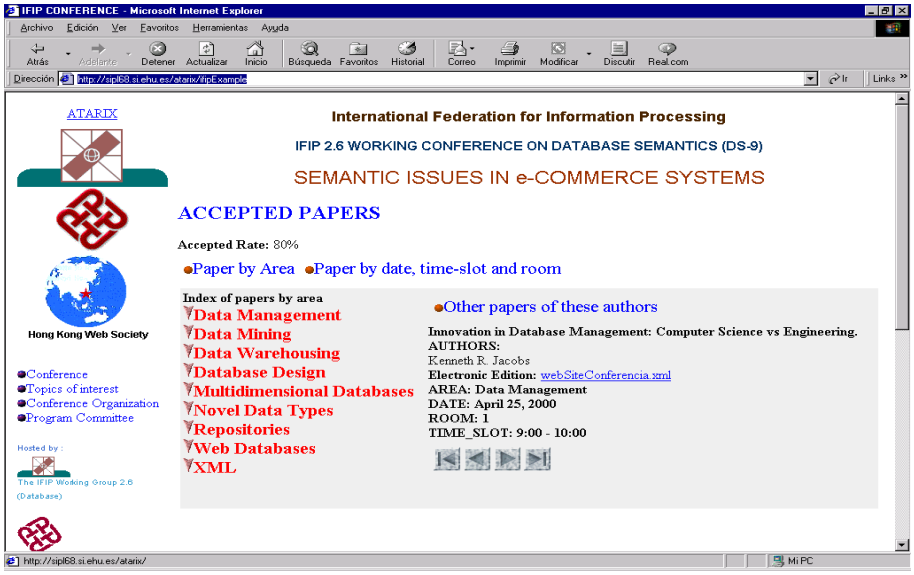

Figure 7. An alternative way to support the navigation from *Conference* to *acceptedPaper*: an index on *area* and then, a scroll along the papers of a selected area.

be one of the values of the *propertyName* element; *"between"* which requires the user to provide the start and end of an interval within which the value of the searched property must be; and *"like"* which requires the user to provide a pattern for searching a particular value.

If the set of destination nodes is low, the designer can choose to traverse the whole set right away. The *CONTENT* element describes this alternative. Notice, that every traversal should end with a *CONTENT* element as this is the only way to reach the content nodes. An alternative is to show the destination nodes one by one rather than the whole set together. In this case, the *CONTENT* element has a *SCROLL* sub-element. Figure 7 illustrates an example: papers are first indexed by *AREA*. As the number of papers within an area can be potentially large, the designer chooses to presented them using a scroll rather than the whole set at once. This is specified as follows:

```
<LINK title="Papers by Area" from="/CONFERENCE to="ACCEPTED_PAPER">
  <INDEX order="1" title="Index of papers by area"
         couplingMode="embedReplacingOnRequest">
    <INDEX_PROPERTY propertyName="AREA"/>
  </INDEX>
  <CONTENT order="2" couplingMode="embedReplacingOnRequest">
    <SCROLL first="yes" previous="yes" next="yes" last="yes"/>
  </CONTENT>
</LINK>
```

An index by *AREA* is provided first. Once an area has been selected, a scroll allows to browse along the papers within this area.

Coupling mode. Let's consider you are visiting the *CONFERENCE* node of the content document. By using an index, you move to one of the papers presented at this conference. The question is wether both the index and the paper should be displayed together with the conference content (*embed*), should substitute the conference content (*replace*) or should be rendered in a separate window (*new*)? Furthermore, should these questions be posed separately for the index and the paper node so that the conference node, the index, and the paper node could be coupled in different ways? Moreover, should this navigation option occur on request or take place automatically when loading the origin (on load)? The coupling mode addresses where and when nodes of the content document, indexes, filters and scrolls are coupled during link traversal[3]. The "where" question admits three possible answers: *embed, replace* and *new*. The "when" question can be answered either with *onLoad* or *onRequest.* As these aspects are not completely orthogonal, we decided to support the following coupling mode values: *newOnLoad, newOnRequest, embedOnLoad, embedOnRequest, embedReplacingOnRequest, embedReplacingOnLoad, replaceOneOnRequest* and *replaceAllOnRequest.*

If distinct navigation modes are used in succession (e.g. an index hierarchy, or an index followed by a filter) the coupling mode of the current navigation mode applies to the previous navigation mode. For instance, the page from the conference application shown in figure 5 delivers the origin element (i.e. a *CONFERENCE* element) in the same page than both the indexes and the destination elements (i.e. an *ACCEPTED_PAPER* element). Alternatively, both the indexes and the selected *ACCEPTED_PAPER*s can be presented separately in a different page from conference data. In this case the navigation mode remains the same while the coupling modes are different. Therefore, the designer is able to decide freely how tightly she wants to distribute the content.

4. The presentation model

Both the content and the link definitions have a presentation counterpart which addresses the look and feel of the final layout. Web rendering is mainly achieved through HTML. This language is too low-level, and it does not provide mechanisms for sharing and reuse. Such lack of suitable abstractions obstructs the construction of frameworks for reuse, and hinders maintenace and evolution. Again separation of concerns and increasing the level of abstractions are the strategies to tackle these problems.

[3]Similar concerns also appear in the W3C XML Linking Language (XLink) proposal [14].

Because *AtariX* is based on XML technology, the XSL philosophy is used for content rendering. A prototype-like mechanism is introduced to enhance reuse in XSL templates, and layout design is split into selecting the delivering mechanism (i.e fonts, background, icons and the like) and the distribution of these items along the presentation space (e.g. a page). Each of these concerns is specified separately in *AtariX*.

Delivering mechanism. One of the most often cited advantages of XML is the separation between content and presentation which can be achieved through XSLT style sheets [15]. An XSLT style sheet includes a set of *templates*, each of which contains information for displaying a particular branch of the element hierarchy in the XML document. The associated *match* attribute identifies the specific branch by using an *XPath* expression. For instance, the following template

```
<xsl:template match="/CONFERENCE/WORKSHOP">
  <h1> <xsl:value-of select="NAME"/> </h1>
  <h2> <xsl:value-of select="ORGANIZER"/> </h2>
</xsl:template>
```

matches those branches of the XML document that belong to a *WORKSHOP* element. In this case, rendering *WORKSHOP* element causes the *NAME* and *ORGANIZER* sub-elements to be presented using the HTML tags *h1, h2*.

Distribution concerns. This refers to how items are arranged on a page. It provides a first skeleton of the layout where some general canvas are drawn (e.g. the header, the body and the footer). The content of these canvas is decided separately in another style sheet which in turn, can include some more detailed layout instructions for further refinement. Unlike the style sheets used for presentation purposes, these sheets are only concerned with distribution. Hence, they are referred to as style-free style sheets[4]. A stepwise process is used similar to JSP [7]. We begin by designing a *general page layout* that has to be followed by any page of the site.

Besides this general layout, separate layouts are provided for each kind of role to be displayed, namely, content, single links, indexes, filters and scrolls. The user can override these default layouts and provide their own. Overriding is provided at the instance level, i.e. it is possible to override how a given index or paper is distributed on the canvas, however the rest of the instances follow the default layout.

For each element (i.e. content, links, filters, scrolls) to be rendered, the system applies the most specific template available. This results in a chunk of HTML code that indicates how this elements should be displayed. In the next step, these chunks of HTML are collected and arranged according to a given page layout. The outcome is a unit of delivery, i.e. a complete HTML page.

5. Conclusions

This paper presents how the common distinction between content, navigation and presentation has been realized in *AtariX*, a tool for web-application development built around a set of XML documents. Distinct XML vocabulary has been introduced to describe the requirements of each dimension in a declarative way. Our contention is that both separation of concerns and declarativeness, account for productivity and maintenance gains during the Web application lifecycle. The designer can change any of these dimensions with a minimum impact on the other dimensions.

A distinctive feature of AtariX is its adherence to the XML "way of working". Not only does *AtariX* use XML to describe the models, but it also uses it for model integration (through XML processing instructions), content location (by means of XPath expressions) or for describing the content's data model. In so doing, we aim to facilitate the acceptance of *AtariX* among the increasing number of practitioners already familiarized with XML.

References

[1] R. Bourret. XML and databases at http://www.rpbourret.com/xml/xmlanddatabases.htm.

[2] S. Ceri, P. Fraternali, and S. Paraboschi. Design Principles for Data-Intensive Web Sites. *SIGMOD Record*, 28(1):84–89, 1999.

[3] S. Ceri, Piero Fraternali, and A. Bongio. Web modeling language (WebML): a modeling language for designing Web sites. *Computer Networks*, 33(1-6):137–157, 2000.

[4] E. Van der Vlist. Style-free XSLT Style Sheets at http://www.xml.com/pub/a/2000/07/26/xslt/xsltstyle.html, 2000.

[5] M. Fernandez, D. Florescu, J. Kang, A. Levy, and D. Suciu. Overview of strudel: A website management system. *Networking and Information Systems Journal*, 1(1):115–140, 1998.

[6] P. Fraternali. Tools and Approaches for Developing Data-Intensive Web Applications: a Survey. *ACM Computer Surveys*, 31(3):227–263, 1999.

[7] David Geary. JSP Templates at http://www.javaworld.com/javaworld/jw-09-2000/jw-0915-jspweb.html, 2000.

[8] T. Isakowitz, E.A. Stohr, and P. Balasubramanian. Rmm: A methodology for structured hypermedia design. *Communications of the ACM*, 38(8):34–43, 1995.

[9] F. Paterno and C. Mancini. Model-Based design of interactive applications. *ACM Intelligence*, pages 27–37, Winter 2000.

[10] R. Kalakota M. Robinson. *e-Business: Roadmap for Success*. Addison-Wesley, 1999.

[11] G. Rossi, D. Schwabe, and F. Lyardet. Web application models are more than conceptual models. In P.P. Chen, D.W. Embley, and S.W. Liddle, editors, *World Wide Web and Conceptual Modeling*, pages 193–208, October 1999.

[12] W3c. XML Path Language (XPath) Version 1.0 at http://www.w3.org/tr/xpath.html, 1999.

[13] W3c. XML-Data at http://www.w3.org/TR/1998/NOTE-XML-data-0105, 2000.

[14] W3c. XML Linking Language (XLinking) Version 1.0 at http://www.w3.org/tr/xlink/, 2000.

[15] W3c. XSL Transformations (XSLT) Version 1.0 at http://www.w3.org/tr/xslt/, 2000.