

A Flexible Layout Model for a Web-Based Adaptive Hypermedia Architecture

Peter Brusilovsky
School of Information Sciences
University of Pittsburgh
135 North Bellefield Avenue
Pittsburgh, PA 15260
peterb@mail.sis.pitt.edu

Paul De Bra and Tomislav Santic
Information Systems Department
Eindhoven University of Technology
PO Box 513
NL 5600 MB Eindhoven, the Netherlands
debra@win.tue.nl, tomi@santic.nl

Abstract

The paper presents an attempt to extend a general-purpose adaptive hypermedia (AH) architecture AHA! with flexible and rich interface by re-using some ideas explored in a more specific AH system InterBook. The most popular application (course) created with the old version of AHA! (the hypermedia course 2L670, later called 2L690, at the TU/e) used a single-window/single-frame concept representation which had advantages of its simplicity and clarity for the user. In this course concepts are always represented in a single frameless window. This mechanism is not sophisticated enough to serve as a general-purpose mechanism for concept representation. InterBook on the other hand has a much richer user interface characterized by the use of multiple windows and frames. This approach has more possibilities for representing the course domain, but has a problem of its rigid presentation structure. The author has no possibilities of adapting the user interface to the specific course which means that every course served by Interbook has the same look independent of the course characteristics. To address the lack of user interface possibilities in AHA! we developed the *Layout model* that employs the strong points of Interbook user interface. The dynamic structures of the *Layout model* are easily extendible and give author the power to adapt the user interface to the nature of the course.

1. Introduction

Numerous Web-based adaptive hypermedia systems have been developed within the last 10 years [8, 9, 10, 11]. These systems all have different “look and feel” and offer different ways of adaptation. Yet, behind this diversity an expert can find a reasonably limited set of methods and techniques [1,2]. A major motivation behind AHA! project [4,6] was developing a flexible adaptive hypermedia architecture that can be used for implementing a wide variety of adaptation methods. AHA! was created as an “assembly language” of adaptive hypermedia in the sense that any higher-level adaptation paradigm can be expressed in terms of AHA! and simulated by the AHA! engine. The most recent AHA! version [4] has shown to be very powerful in this respect. As was recently demonstrated, a reasonably advanced adaptation paradigm implemented in InterBook system [3] can be almost completely simulated by AHA! [4, 12]. Yet, with all this power, AHA! is not completely ready to serve as a universal simulator for an arbitrary AH system.

The problem here is that each AH system is characterized not only its unique model of adaptation, but also by its unique interface. While the current version of AHA! is quite ready to simulate the adaptation behavior of any AH system, simulating their interface is a lot more difficult. The AHA! engine operates in a single-window mode - in a good tradition of classic hypertext systems. In contrast, most of modern adaptive Web-based hypermedia systems use

rich multi-frame and multi-window interfaces. InterBook is a good example here. It uses several multi-frame windows (textbook, glossary, and table of contents). An example of InterBook's Textbook and Glossary windows is provided on Figure 1. Other advanced AH systems also use complex multi-frame windows. Creating such multi-frame and/or multi-window interfaces in AHA! requires that the author define the frame structure and use Javascript code to synchronize the presentation in all the frames.

The goal of the project introduced in this paper was to resolve the problem by developing a flexible interface model of AHA! engine. Following the idea of AHA! that can be used to describe a variety adaptation functionalities, we wanted to develop an interface model that is used to describe a variety of Web adaptive hypermedia interfaces. The primary goal of our project was reasonably modest: we wanted to extend the AHA! engine to the extent where it can simulate the InterBook adaptation mechanism and its multi-frame interface. In doing so, we wanted to avoid narrow-minded solutions and hacks (like the Javascript hack previously used with AHA!) developing a reasonably universal approach that can be used to implement the InterBook interface along with many other interfaces. This paper presents the first results of our work. To introduce the background for our work, we start with a brief introduction of the InterBook and AHA! interfaces. After that, we present our Layout Model that extends AHA! and demonstrate how it can be used to simulate the InterBook interface in AHA!.

2. The InterBook Interface Paradigm

InterBook has two main kinds of windows - a Textbook window (left on Figure 1) and a Glossary Window (right on Figure 1). These windows correspond to two major kinds of information items supported by InterBook - a book page and a domain knowledge concept. Each window in InterBook can include multiple links to concepts and pages. A click on any page link causes the linked page to be loaded in the Textbook window. A click on any concept link causes the information about the linked concept to be loaded in the glossary Window.

Despite of its complicated interface, InterBook attempted to support a simple metaphor - one window shows one and only one information item - i.e., a textbook window shows exactly one page of a textbook at a time. While each of these two windows includes several frames, there are considered not as independent windows, but as multiple views on the same concept or page. For example: the text frame (bottom left) presents the text of the page; the navigation bar (top) presents the location of the current page among its ancestors and siblings, and the concept bar (bottom right) presents prerequisite and outcome concepts for the current page. All four frames of the textbook window are updated at the same time. Technically, a link to a textbook page is calling a whole *page frameset* to be loaded into the textbook window. This frameset, in turn, pulls several frames associated with the requested page. The frameset approach is simple to understand and also works well with most browsers' standard way of navigation using back and forward buttons and history.

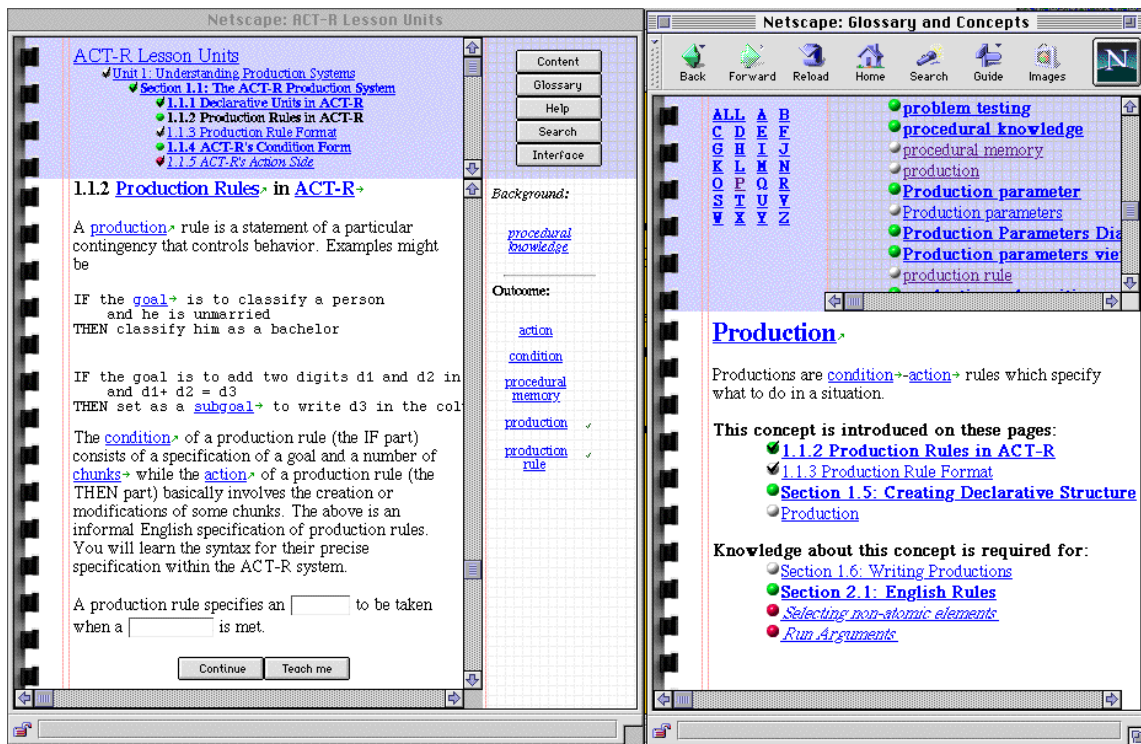


Fig. 1. InterBook interface.

3. The AHA! Interface Paradigm

AHA! was initially created to add adaptation to the course on hypermedia at the Eindhoven University of Technology (<http://www.wis.win.tue.nl/2L690/>). This course predates the general availability of frames in browsers. The course was therefore written using a single frame layout. The browser showed one course page at a time, with adapted links and conditionally included fragments. AHA! also added an optional header (with a progress report) and footer (with copyright statement). Header and footer were created by the author as html fragments. In Figure 3 a page with header can be seen (albeit embedded in the new Interbook-like multi-frame environment). Multi-frame applications are possible in AHA!. Figure 2 shows part of the multi-frame interface paradigm used in a course on graphical user interfaces.

In order to make this interface work in AHA!, every page must include the following piece of Javascript code:

```
<script language="JavaScript">
  parent.frames[0].location="content.xml"
</script>
```

The result is that when a link to a page is followed the leftmost frame is reloaded. It contains the "content.xml" file, which is a navigation menu in which submenus are conditionally shown, based on which page is displayed in the rightmost frame. The access to a page and the subsequent access to the menu are treated separately by AHA!. Whereas in Interbook following a link requests a complete frameset from the server in AHA! following a link requests a page, to be shown in a complete browser window or inside a frame. The AHA! engine does not "know" about a possible use of frames.

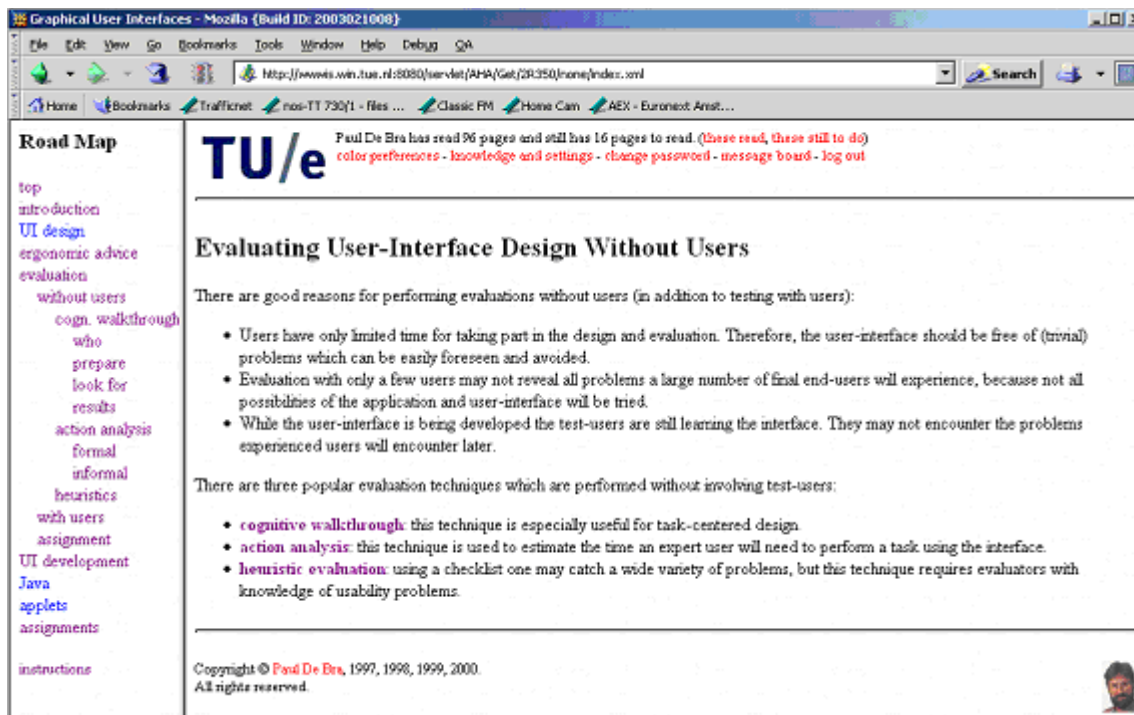


Fig. 2. AHA! multi-frame interface.

4. The View-Based Layout Model

The View-Based Layout Model is the new way in AHA! to present concepts (pages) to the user that was developed to address the lack of user interface possibilities in the earlier versions of the architecture. The *Layout model* combines the strong points of InterBook's rich user interface with the flexibility and customization style that are typical for the AHA!-architecture. This model allows every AH system developer to adapt the user interface to the course nature (without the need for the above mentioned Javascript hack).

To provide a high level of flexibility, the Layout Model was designed as a three-level interface model that is based on the concepts *views*, *viewgroups* and *layouts*. In brief, *views* are considered as atomic interface elements. Views can be grouped in *viewgroups*. One or more *viewgroups* form a *layout*. These concepts are presented in more detail below.

4.1 Views

Views are pieces of information about the course domain. They usually represent some relevant information about the active concept (the concept the user is viewing at the moment). A view can also represent some static information about the course domain. Views are used as pre-fabricated building blocks to construct the user interface for some specific course. Internally views are simply Java classes that generate HTML pages using underlying AHA! data structures. To present a concept to the user the system uses a set of predefined views. These predefined views can be further customized by the author of the course to develop an interface that meets the needs of the course. The author defines and customizes a view using an XML-based description language like the following:

```

<view name="v5" type="ToolboxView" title="Toolbox"
  background="IBookbluesq.bmp" params="leftspace=70">
  <secwnds>
    <secwnd link="TOC" viewgroup="TOC" img="ContentBtn.bmp"/>
    <secwnd link="Glossary" viewgroup="Glossary"
      img="GlossaryBtn.bmp"/>
  </secwnds>
</view>

```

At the moment we have already implemented a number of relatively simple basic views. The configuration of these views consists of setting the background picture, the title or changing the page margin to make the view more readable. It is possible however to implement much more complex views that will offer much more tuning possibilities to the author. We are considering parameters that will influence the content of the view page and not only the shape of it.

A view usually displays some information about the active concept including links to other relevant concepts. However a view can also contain links to other views which will offer more information to the user about the active concept. Following one of these links will result in displaying a new set of views. Views that are used directly to represent different aspects of a concept are called *primary views*. Views that present some supplementary information are called *secondary views*. These views are not visible until they are triggered by a link in one of the primary views. The author of the course will usually choose the most important views as primary views and less important views as secondary views. The connection between primary and secondary views can be specified by the author of the course in the XML view structures presented above. In the presented example *ToolboxView* can trigger two secondary viewgroups: *Table of Content* and *Glossary*.

4.2 Viewgroups and Layouts

As already said views are the building blocks for constructing concept representation. Views can be grouped in *viewgroups*. In HTML terms a *viewgroup* corresponds to an independent window and a view corresponds to a page that can be shown in a separate window or in a window frame. A set of *viewgroups* forms a *concept layout*, which is used to present a concept. Practically, it means that different aspects of a concept can be presented in several synchronized windows.

We assume that the system may have more than one *type* of concepts (pages). For example, InterBook has a *textbook page* and a *glossary concept*

that are both concepts in terms of the AHA! architecture. We also assume that an author of an adaptive course may want different types of concepts to be presented differently (this is what happens in InterBook). To support this possibility, our Layout model allows an author to define several layouts. Each concept type has to be associated with one of the layouts. Presenting concepts of the same type always in the same way (using the same layout) contributes to the user confidence in the system and avoids confusion. Links to the concepts of the same type are also annotated in the same way for obvious reasons.

The following XML structure is an example of a layout definition for two layouts that we use to simulate an InterBook style user interface:

```

<layoutlist>
  <layout name="page_c_layout" trigger="MAIN">
    <viewgroup name="MAIN" wndOpt="width=800,height=600">
      <compound framestruct="rows=20%,*" border="0">
        <compound framestruct="cols=*,130" border="0">
          <viewref name="v1" />
          <viewref name="v5" />
        </compound>
        <compound framestruct="cols=*,130" >
          <viewref name="v3" />
          <viewref name="v2" />
        </compound>
      </compound>
    </viewgroup>
    <viewgroup name="TOC"
wndOpt="resizable=1,toolbar=1,width=300,height=400">
      <viewref name="v1"/>
    </viewgroup>
    <viewgroup name="Glossary" secondary="true"
wndOpt="width=600,height=500">
      <viewref name="v4"/>
    </viewgroup>
  </layout>
  <layout name="abst_c_layout" trigger="Glossary" >
    <viewgroup name="Glossary" wndOpt="toolbar=1,width=600,height=500">
      <viewref name="v4"/>
    </viewgroup>
  </layout>
</layoutlist>

```

We have defined two layouts each associated with one of the two concept types that we use in AHA! at the moment: page concepts and abstract concepts. As can be seen in the example above each layout consists of a set of viewgroups which contain pointers to predefined views. Viewgroups use *compound* elements to define the place of each of the views within the window. For each viewgroup the author of the course can also define window options for the window in which the viewgroup is placed. The layout structure of layout '*page_c_layout*' above corresponds to the screen presented in Figure 3.

This layout consists of four primary views grouped into one viewgroup, which is shown in the figure, and two secondary views (Glossary and Table of Content) which can be triggered by the buttons in the ToolboxView (upper right corner).

Changing the XML configuration structures will change the layout associated with a certain concept type. The following example of an XML configuration structure uses the same views for the same concept type but grouped in a different way:

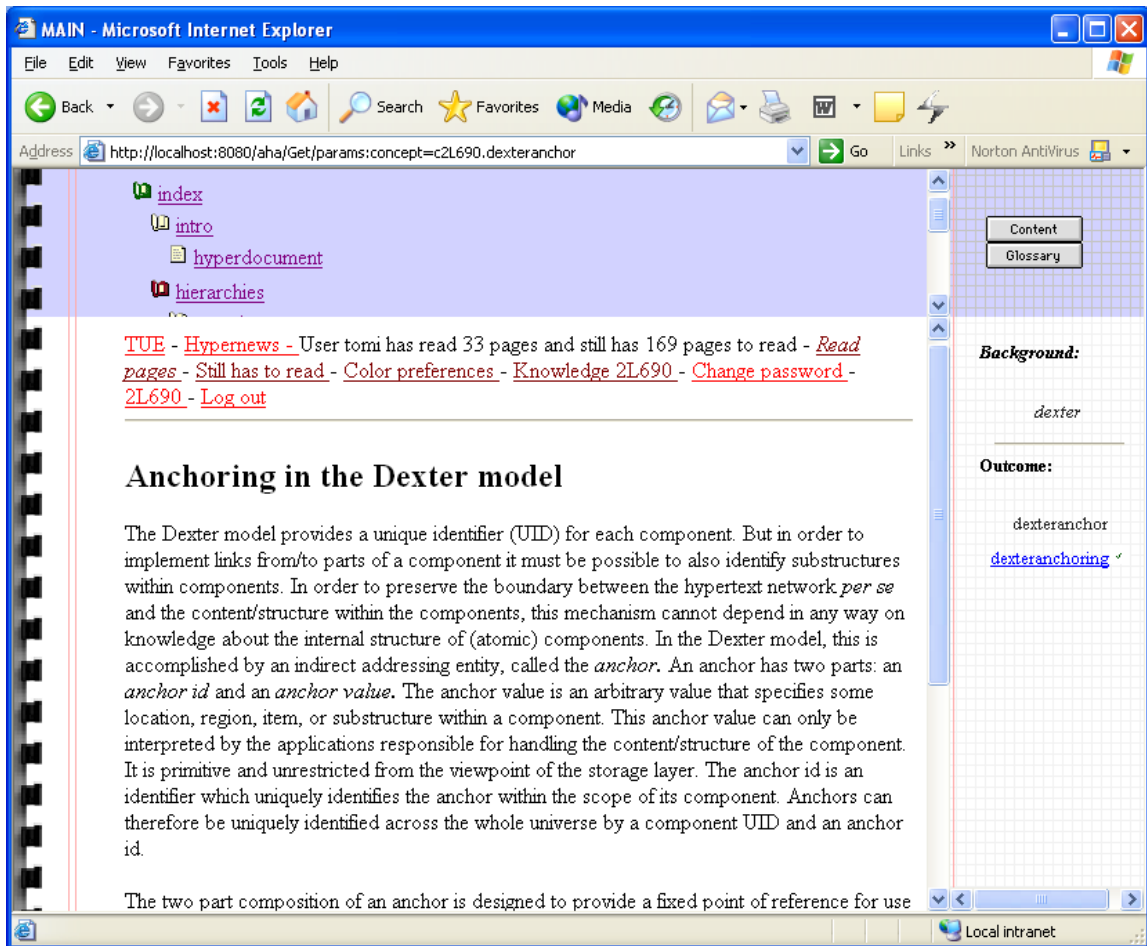


Fig. 3: InterBook style concept layout for 'page' concepts

```

<layout name="page_c_layout" trigger="MAIN">
  <viewgroup name="MAIN"
    wndOpt="status=1,menubar=1,resizable=1,toolbar=1,width=800,height=6
00">
    <compound framestruct="cols=200,*" >
      <compound framestruct="rows=*,85" >
        <viewref name="v1" />
        <viewref name="v5" />
      </compound>
      <viewref name="v3" />
    </compound>
  </viewgroup>
  <viewgroup name="Conceptbar"
    wndOpt="status=1,menubar=1,resizable=1,toolbar=1,width=300,height=4
00">
    <viewref name="v2" />
  </viewgroup>
  <viewgroup name="Glossary" secondary="true"
    wndOpt="status=1,menubar=1,resizable=1,toolbar=1,width=600,height=5
00">
    <viewref name="v4" />
  </viewgroup>
</layout>

```

The corresponding screen layout for the XML configuration structure above is shown in figure 4.

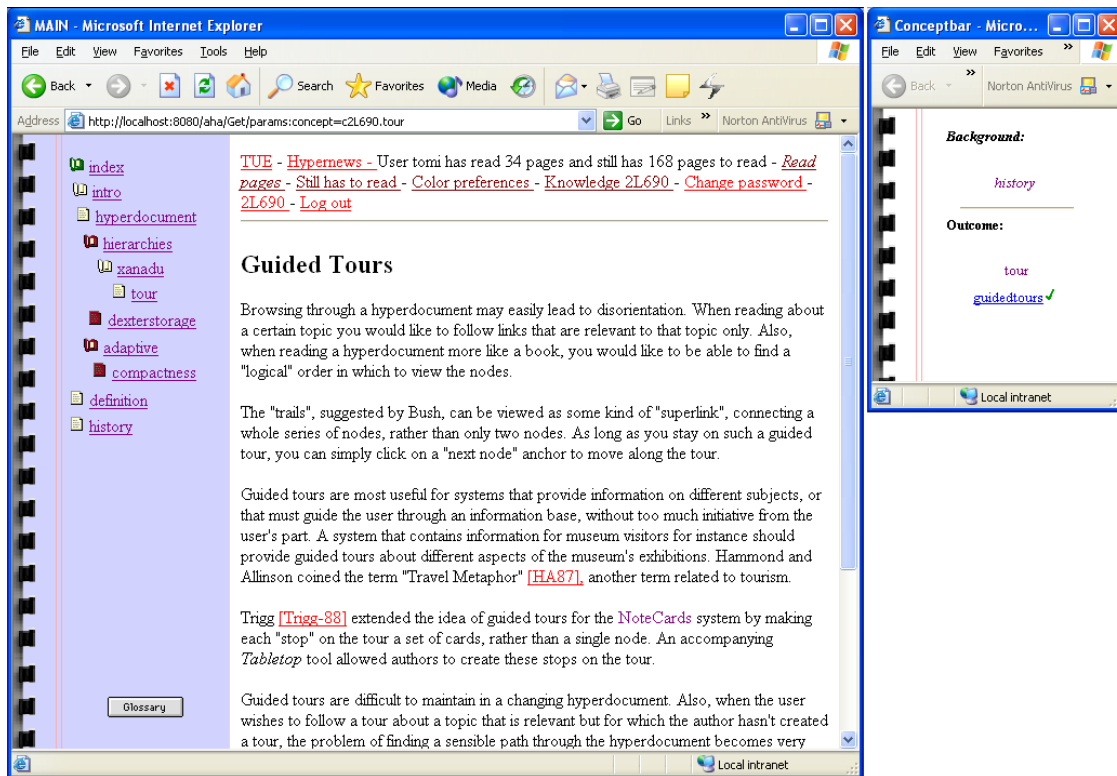


Fig. 4. second version of ‘page’ concepts layout

In this version of the layout associated with page concepts there are two primary viewgroups (MAIN and Conceptbar) and one secondary viewgroup (Glossary). Viewgroup MAIN consists of three views (MainView, Table of Content and Toolbox) and the Conceptbar viewgroup contains one view (ConceptbarView). Button ‘Glossary’ in the Toolbox view triggers the display of the secondary viewgroup *Glossary*.

5. Conclusions and Future work

The new AHA! layout model offers versatile user interface possibilities and brings AHA! one step closer to its main goal of being a generic Adaptive Hypermedia environment for all kinds of Adaptive Hypermedia applications. View based concept presentation is extremely flexible and gives a course author the power to adapt the user interface to the needs of the course. Internally views are Java objects with one task: extracting data from AHA! data structures and generating HTML pages from these data. In the future we are planning to extend the user interface adaptation possibilities by introducing the total data-presentation separation. We are thinking of giving the author the opportunity of implementing his/hers own views, in addition of using a set of predefined views. If the internal static AHA! data structures would be saved as XML files the author could use any standard XSLT editor to implement views as XSLT files which could extract data from XML formatted data structures. This model would give the author the possibility to represent the data in any desirable way without being dependent on already implemented views.

References

1. Brusilovsky, P. (1996) Methods and techniques of adaptive hypermedia. In P. Brusilovsky and J. Vassileva (eds.), *User Modeling and User-Adapted Interaction 6* (2-3), Special Issue on Adaptive Hypertext and Hypermedia, 87-129.
2. Brusilovsky, P. (2001) Adaptive hypermedia. *User Modeling and User Adapted Interaction 11* (1/2), 87-110, also available at <http://www.wkap.nl/oasis.htm/270983>.
3. Brusilovsky, P., Eklund, J., and Schwarz, E. (1998) Web-based education for all: A tool for developing adaptive courseware. *Computer Networks and ISDN Systems (Proceedings of Seventh International World Wide Web Conference, 14-18 April 1998)* 30 (1-7), 291-300.
4. De Bra, P., Aerts, A., Smits, D., and Stash, N. (2002) AHA! Version 2.0: More Adaptation Flexibility for Authors. In: M. Driscoll and T. C. Reeves (eds.) *Proceedings of World Conference on E-Learning, E-Learn 2002, Montreal, Canada, October 15-19, 2002, AACE*, pp. 240-246.
5. De Bra, P., Brusilovsky, P., and Houben, G.-J. (1999a) Adaptive Hypermedia: From Systems to Framework. *ACM Computing Surveys* 31 (4es), also available at http://www.cs.brown.edu/memex/ACM_HypertextTestbed/papers/25.html.
6. De Bra, P. and Calvi, L. (1998) AHA! An open Adaptive Hypermedia Architecture. In P. Brusilovsky and M. Milosavljevic (eds.), *The New Review of Hypermedia and Multimedia 4, Special Issue on Adaptivity and user modeling in hypermedia systems*, 115-139.
7. De Bra, P., Houben, G. J., and Wu, H. (1999b) AHAM: A Dexter-based Reference Model for Adaptive Hypermedia. In: *Proceedings of 10th ACM Conference on Hypertext and hypermedia (Hypertext'99), Darmstadt, Germany, February 21 - 25, 1999, ACM Press*, pp. 147-156.
8. Grigoriadou, M., Papanikolaou, K., Kornilakis, H., and Magoulas, G. (2001) INSPIRE: An INtelligent System for Personalized Instruction in a Remote Environment. In: P. D. Bra, P. Brusilovsky and A. Kobsa (eds.) *Proceedings of Third workshop on Adaptive Hypertext and Hypermedia, Sonthofen, Germany, July 14, 2001, Technical University Eindhoven*, pp. 13-24.
9. Henze, N. and Nejdil, W. (2001) Adaptation in open corpus hypermedia. In P. Brusilovsky and C. Peylo (eds.), *International Journal of Artificial Intelligence in Education 12* (4), Special Issue on Special Issue on Adaptive and Intelligent Web-based Educational Systems, 325-350, also available at http://cbl.leeds.ac.uk/ijaied/abstracts/Vol_12/henze.html.
10. Melis, E., Andrès, E., Büdenbender, J., Frishauf, A., Goguadse, G., Libbrecht, P., Pollet, M., and Ullrich, C. (2001) ActiveMath: A web-based learning environment. In P. Brusilovsky and C. Peylo (eds.), *International Journal of Artificial Intelligence in Education 12* (4), Special Issue on Special Issue on Adaptive and Intelligent Web-based Educational Systems, 385-407.
11. Weber, G. and Brusilovsky, P. (2001) ELM-ART: An adaptive versatile system for Web-based instruction. In P. Brusilovsky and C. Peylo (eds.), *International Journal of Artificial Intelligence in Education 12* (4), Special Issue on Adaptive and Intelligent Web-based Educational Systems, 351-384, also available at http://cbl.leeds.ac.uk/ijaied/abstracts/Vol_12/weber.html.
12. Wu, H., De Kort, E., and De Bra, P. (2001) Design Issues for General Purpose Adaptive Hypermedia Systems. In: *Proceedings of Twelfth ACM Conference on Hypertext and Hypermedia (Hypertext 2001), Aarhus, Denmark, August 14-18, 2001, ACM Press*, pp. 141-150.