AHA! meets Interbook, and more…

Paul De Bra[*] and Tomislav Santic
Department of Computing Science
Eindhoven University of Technology (TU/e)
PO Box 513, Eindhoven
The Netherlands
debra@win.tue.nl, tomi@santic.nl

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

Abstract: The AHA! system (De Bra & Calvi, 1998, De Bra et al, 2002) has been repeatedly extended over the past few years, focusing on adaptation flexibility. This has resulted in a powerful adaptation engine, but little support for creating adaptive applications. AHA! provides tools for defining the conceptual structure and the adaptation of an application, but leaves the presentation and additional support tools up to the author of that application. Interbook (Brusilovsky et al, 1998) on the other hand is a simple environment for creating and serving adaptive textbooks, with a rich user interface characterized by the use of multiple windows and frames. Authors write an annotated Microsoft Word file, which is translated to a series of files used by Interbook. This paper presents an extension of AHA! that enables a high-level specification of the presentation (layout) of an AHA! application. We illustrate this extension through a powerful demonstrator: an Interbook to AHA! compiler. The source format for Interbook is translated to AHA! with the new Layout model. 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 application. AHA! can thus "emulate" not just Interbook but other adaptive environments as well.

Introduction

Numerous Web-based adaptive hypermedia systems have been developed within the last 10 years (Grigoriadou et al, 2001, Henze & Nejdl, 2001, Melis et al, 2001, Weber & Brusilovsky, 2001). These systems all have a 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 (Brusilovsky, 1996, Brusilovsky, 2001). A major motivation behind the AHA! project (De Bra & Calvi, 1998, De Bra et al, 2002) 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 (De Bra et al., 2002) was shown to be very powerful in this respect. The reasonably advanced adaptation paradigm implemented in the InterBook system (Brusilovsky et al, 1998) can be simulated by AHA! (De Bra et al, 2002, Wu et al, 2001). However, recreating the layout/presentation offered by InterBook would be very laborious, as until now AHA! did not offer any support for multi-frame presentations. (Multi-frame applications are possible, as demonstrated in a course offered at the TU/e, but the synchronization between the frames has to be programmed by the author, using JavaScript.) The AHA! engine responds to an HTTP (get) request by returning one HTML document to the browser. When using multiple frames, the requests to load HTML documents in the different frames are treated by AHA! as completely independent requests. In other words, the AHA! engine does not "know" that multiple frames exist in the application. In contrast, many modern adaptive Web-based hypermedia systems use rich multi-frame or even 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 in (Fig. 1).

The goal of the project introduced in this paper was to resolve the problem by developing a flexible interface model for the AHA! engine. Following the idea of AHA! that can be used to describe a variety of adaptation functionalities, we wanted to develop an interface model that is used to describe a variety of adaptive Web-based hypermedia interfaces. The primary goal of our project was reasonably modest: we wanted to extend the AHA! engine to enable it to 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.

[*] Also at the "Centrum voor Wiskunde en Informatica" in Amsterdam, and the University of Antwerp, Belgium.

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!.

The InterBook Interface Paradigm

InterBook has two main kinds of windows - a Textbook window, left on (Fig. 1), and a Glossary Window, right on (Fig. 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.

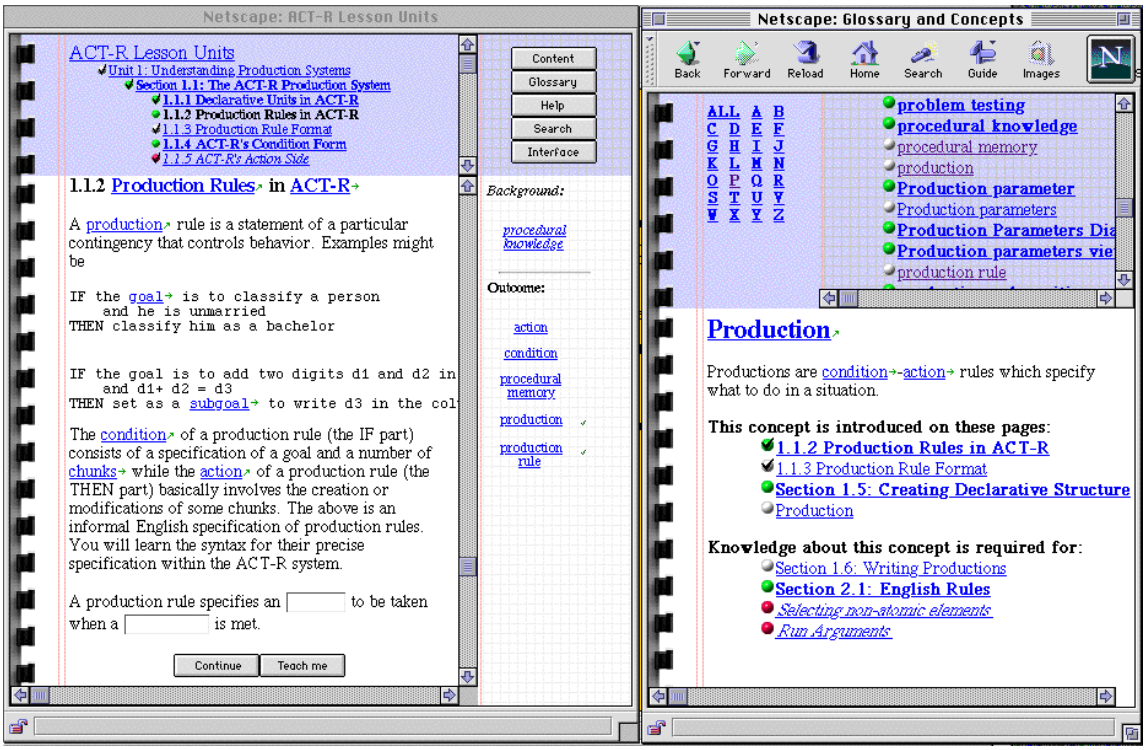

Figure 1: InterBook interface.

The AHA! Interface Paradigm

AHA! was initially created to add adaptation to the course on hypermedia at the Eindhoven University of Technology (currently available at http://wwwis.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. Multi-frame applications are possible in AHA!. (Fig. 2) shows part of the multi-frame interface paradigm used in a course on graphical user interfaces, also at the TU/e.

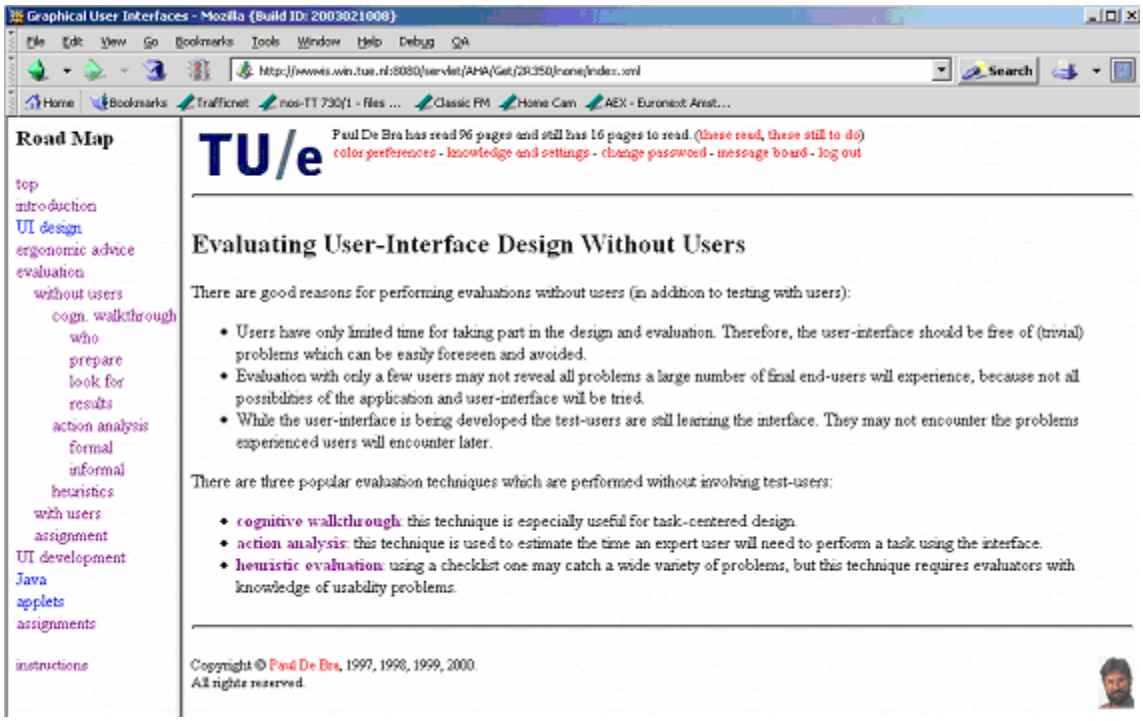

Figure 2: AHA! multi-frame interface.

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” 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.

The View-Based Layout Model

The View-Based Layout Model is the new way in AHA! to present concepts (pages) to the user. It was developed to address the lack of user interface possibilities in the earlier versions of the AHA! 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 adaptive courseware 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.

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 (frames) 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 (and appear in secondary viewgroups). These views and viewgroups 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.

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 browser window or in an HTML frame within a window. A set of viewgroups forms a concept layout, which is the entire presentation of 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 (Fig. 3).

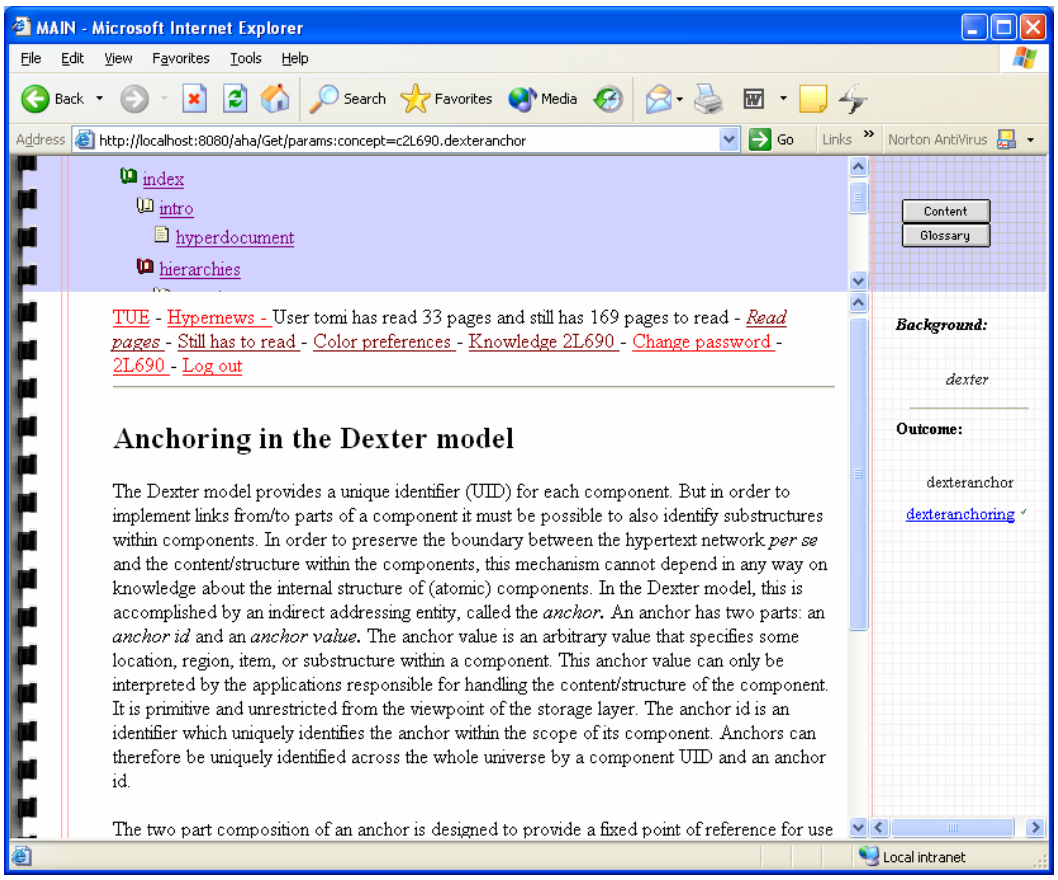

Figure 3: InterBook style concept layout for 'page' concepts

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:

```
<layout name="page_c_layout" trigger="MAIN">
   <viewgroup name="MAIN"
      wndOpt="status=1,menubar=1,resizable=1,toolbar=1,width=800,height=600">
      <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=400">
      <viewref name="v2"/>
   </viewgroup>
   <viewgroup name="Glossary" secondary="true"
      wndOpt="status=1,menubar=1,resizable=1,toolbar=1,width=600,height=500">
      <viewref name="v4"/>
   </viewgroup>
</layout>
```

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

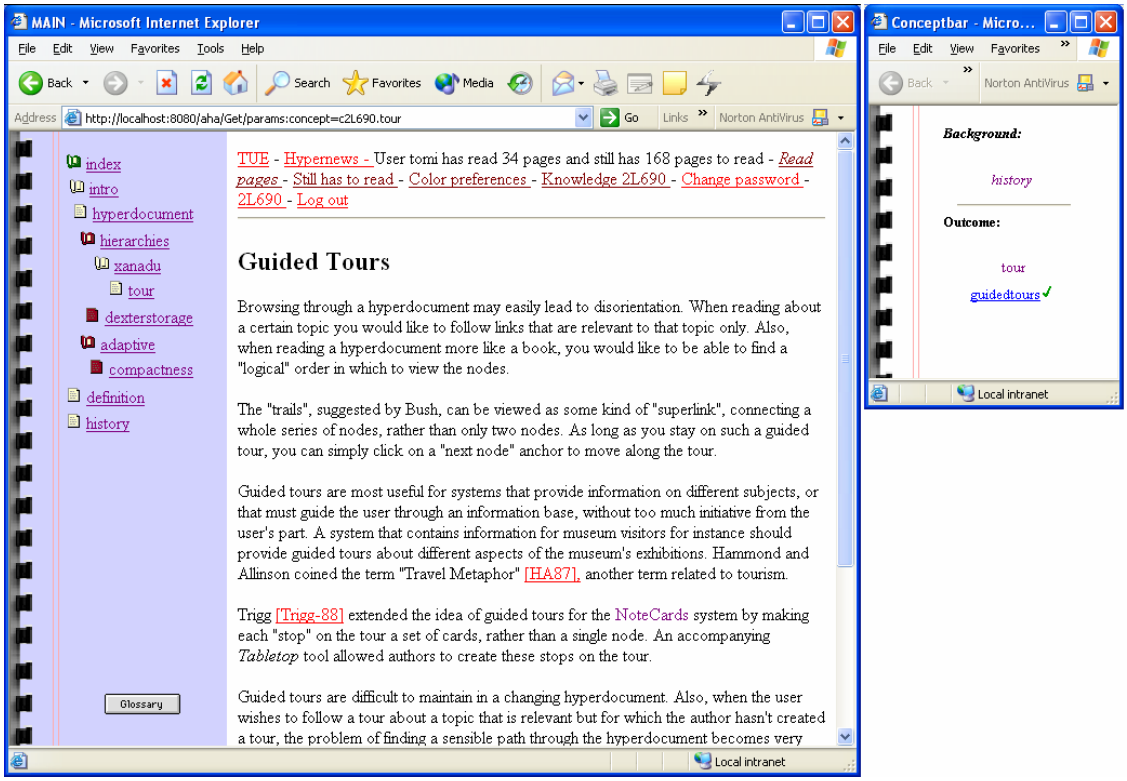

Figure 4: second version of 'page' concepts layout

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.

The Interbook to AHA! Compiler

The final step in our attempt of bringing AHA! and Interbook together is the implementation of Interbook to AHA! compiler. There are several reasons that make this step extremely important:

- Testing the flexibility of our layout model by simulating real courses already offered by other Adaptive Hypermedia systems (in this case Interbook);
- Testing the correctness of data extracted by views. We can compare the course data served by Interbook with data served by AHA!;
- Achieving of total simulation of Interbook by AHA!. AHA! can serve real Interbook courses that also look like Interbook;
- Authoring of Interbook courses is much easer than authoring of AHA! courses. Implementation of a bridge between these two systems can simplify the AHA! authoring mechanism. The author can use the Interbook authoring mechanism (using Microsoft Word and tools to generate HTML from that) to implement a course and then use the Interbook to AHA! complier to convert the course to AHA! format. We must note that this authoring process does not use the full power of AHA!. In particular it only uses link adaptation, not content adaptation in the form of the conditional inclusion of fragments. (But that can be added later.)

Paradigm translation

The most important part of the Interbook to AHA! compiler is the translation of the Interbook concept paradigm to the AHA! concept paradigm. These two paradigms differ in some basic aspects which make it difficult to serve Interbook courses using the AHA! engine. The Interbook paradigm consists of text pages, also called sections, and glossary concepts. Text pages are presented in a Text Window and glossary concepts are presented in a Glossary Window. AHA! on the other hand sees everything as a "concept". We introduced *concept types* to the AHA! concept paradigm. This turned to be a very simple and versatile solution for our problem. Each concept is of some type and each concept type is associated with different layout. This means that each concept type can be represented in a different way, depending on the associated layout, which is exactly what we need to simulate Interbook courses. AHA! does not have a predefined set of concept types. The author of a course can define any desirable number of concept types and represent every concept type using different layout. The connection between concept types and the layout model, which can be established using small XML configuration files, offers great flexibility and possibility to simulate the user interface of almost every existing Adaptive Hypermedia system.

To get back to the Interbook concept paradigm, our Interbook to AHA! compiler generates three kinds of concepts to simulate Interbook courses:

1. Items (simulation of Interbook concepts presented in a Glossary Window)
2. Sections (simulation of Interbook text pages presented in a Text Window having child nodes)
3. Leafs (simulation of Interbook text pages presented in a Text Window without child nodes)

All these concept types are associated with different layouts. The difference between Sections and Leafs is very small. They use layouts that are almost the same with the difference that the Sections layout shows the child nodes of the active concept and the Leafs layout does not.

Compiler Input/Output

The Interbook to AHA! compiler uses special Interbook files as input and produces AHA! formatted XML files as output. Interbook course files are valid HTML documents that contain Interbook specific codes. These codes are used to connect sections (text pages) and glossary concepts. Every section has a set of prerequisite concepts (concepts that are required to be known before reading the section) and a set of outcome concepts (concepts that are introduces by the section).

AHA! courses are saved in a different way. AHA! uses XML structures to save concept data and separate XHTML files are used for the resources. AHA! XML concept structures are much more complex than Interbook concept relations. Interbook uses two kinds of relations between Sections and Glossary concepts: 'is prerequisite' and 'is outcome'. AHA! on the other hand uses expressions to implement different kinds of relationships. These expressions can be of any form as long as they are syntactically valid. We use the following expressions to simulate Interbook concept relationships and Interbook behavior:

- For each prerequisite concept of each section:
  '**Conceptname.*knowledge*>=(1/3*100)**' for AHA requirement relationship which simulates Interbook 'is prerequisite' relationship
- For each outcome concept of each section:
  '**if (required) Conceptname.*knowledge*+1/3*(100-Conceptname.*knowledge*)**
  else Conceptname.*knowledge*+1/6*(100-Conceptname.*knowledge*) '
  for AHA Condition-Action rules which simulate Interbook 'is outcome' relationship

(Fig. 5) shows the course data transformation from Interbook format to AHA! format.

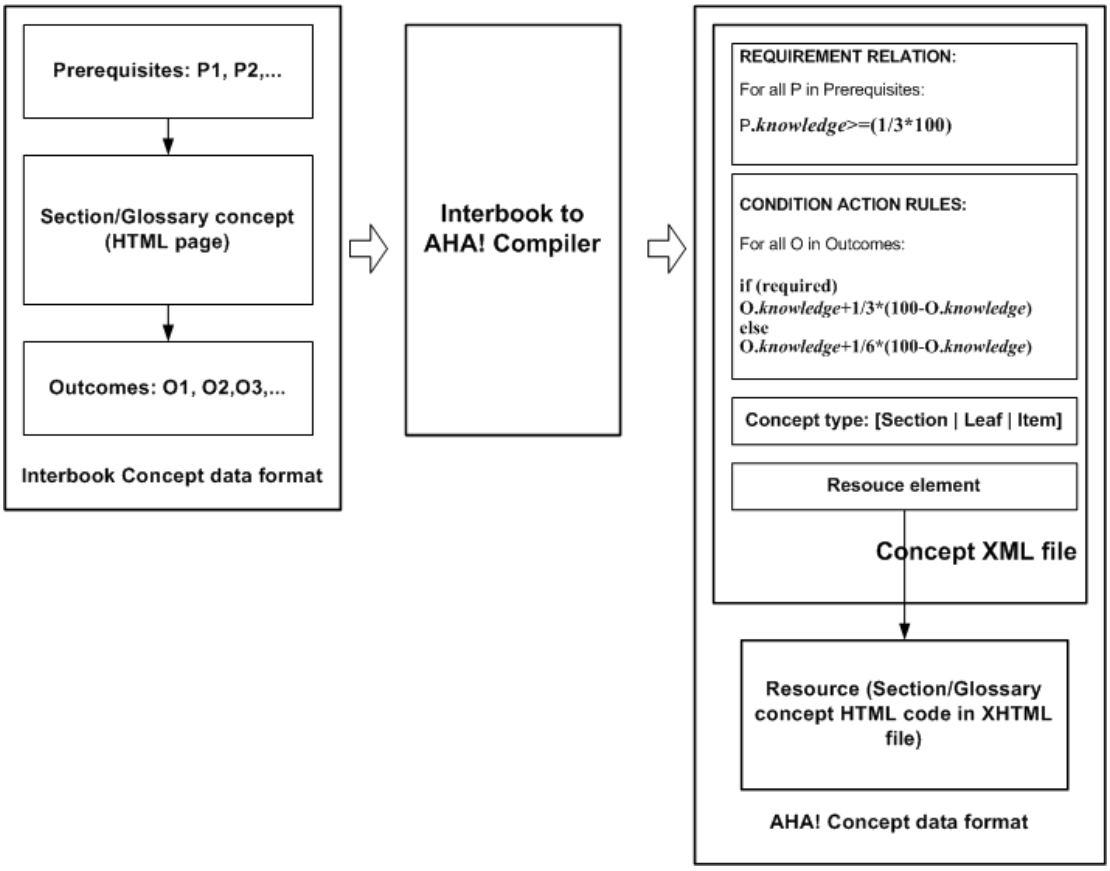

Figure 5: Interbook to AHA! concept data transformation

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 to 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.

The Interbook to AHA! compiler is an important step towards improving the usability of both Interbook and AHA!. Authoring adaptive textbooks for Interbook is relatively easy as it is done in Microsoft Word (together with some conversion to HTML). Compiling the applications to (the new) AHA! makes Interbook independent of the specific server environment required by Interbook (a Lisp-based Webserver) and opens up the possibility to add content adaptation (conditionally included text fragments) to Interbook applications.

Acknowledgement

The development of AHA! was made possible through a grant of the NLnet Foundation, and the hospitality of the University of Pittsburgh.

References

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.

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.

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.

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.

De Bra, P., Brusilovsky, P., and Houben, G.-J. (1999a) *Adaptive Hypermedia: From Systems to Framework.* ACM Computing Surveys 31 (4es): http://www.cs.brown.edu/memex/ACM_HypertextTestbed/papers/25.html.

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.

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.

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.

Henze, N. and Nejdl, 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, http://cbl.leeds.ac.uk/ijaied/abstracts/Vol_12/henze.html.

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.

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: http://cbl.leeds.ac.uk/ijaied/abstracts/Vol_12/weber.html.

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.