Creating Adaptive Textbooks with AHA! (An Interactive RoundTable)

Paul De Bra, Natalia Stash, David Smits
Department of Computing Science
Eindhoven University of Technology (TU/e)
PO Box 513, Eindhoven, The Netherlands
{debra,nstach,dsmits}@win.tue.nl

Abstract: AHA! stands for the Adaptive Hypermedia Architecture, an adaptive authoring and delivery platform developed as part of the Adaptive Hypermedia for All (or AHA!) project. (See http://aha.win.tue.nl/.) Adaptation in on-line textbooks makes it possible for learners to study the textbook in different ways without encountering difficulties. Link hiding and annotation guide users towards subjects they are "ready" to study. Adaptive sequencing or sorting helps them decide on a reading order for pages that teach them about a given concept. Conditional inclusion of fragments and stretchtext make it possible to provide additional or prerequisite explanations when needed or desired. In this roundtable (paper) we discuss the authoring process for adaptive textbooks created with AHA!. We cover the different authoring tools for concepts, concept relationships (like prerequisites) and adaptive object inclusion (for additional explanations). We also cover the process of completing an application by creating content pages in standard XHTML (or in XHTML+SMIL or SMIL 2.0) and by setting up a server. AHA! applications can also be created using other authoring tools, including Interbook (Brusilovsky et al, 1998) and MOT (Cristea et al, 2003), but this is described elsewhere.

Introduction and Background

Adaptive systems have started to emerge in the early 1990's, with a strong emphasis on adaptation in educational applications. The overview paper by Brusilovsky (Brusilovsky, 1996) marked a turning point from mainly platform-dependent systems to Web-based environments. The updated overview (Brusilovsky, 2001) reflects this move towards Web-based systems. The work on the AHA! system (the Adaptive Hypermedia Architecture) started in 1996, and is still ongoing as part of the Open Source Adaptive Hypermedia for All project, partially funded by the NLnet Foundation. AHA! is attracting interest from many institutes and development effort from some, to add new features and tools, see e.g. (Cini et al, 2002), (Romero et al, 2002, 2003).

The main goal of AHA! is to provide a general-purpose adaptive hypermedia platform, consisting of an authoring and a delivery part. In this paper we will concentrate on the authoring part of AHA! (but provide basic information on setting up the server part as well). AHA! concentrates on authoring at the *conceptual* level. Of course an author has to create (or retrieve and provide) pages or other content material, but AHA! first of all links pages or resources to *concepts*, and the authoring is then mostly done on *concepts* and *concept relationships*. Adaptation is specified at this level, and automatically translated (by the authoring tools) to lower level adaptation of links and content. In Brusilovsky's terms (Brusilovsky 1996 and 2001) the AHA! delivery system offers a rich (and recursive) form of *conditional inclusion of fragments* and *link hiding* or *link annotation*.

Contrary to most other adaptive hypermedia (learning) systems AHA! enforces very little in the adaptation or the presentation. We previously illustrated the presentation flexibility by creating a layout that closely resembles Interbook (De Bra et al, 2003b). The *layout model* allows detailed control over the presentation format. Concept relationship *templates* allow for the definition of adaptation to arbitrarily many different types of relationships, not just to *prerequisites*. Conditional *object inclusion* (De Bra et al, 2003a) is a powerful means to add or select details or explanations (without redundancy) or to select between media types. *Presentation stability* enables adaptation without confusing the learner by presentations or information that changes on her when revisiting pages.

This paper is organized as follows: we first show the overall architecture and typical aspects of authoring the *content* of an adaptive textbook for AHA!. We then introduce the authoring tools for creating the *conceptual structure* of an application. After this main topic we briefly describe some special tools AHA! offers for educational applications, including progress reports, forms for end-user manipulation of the user model, and multiple-choice tests. We conclude with a short description of advanced features of AHA! (mentioned above).

Overall AHA! Architecture

The core of AHA! is a Java-servlet-based Web-server extension that performs adaptation to local or external pages while serving these pages to the end-user. The adaptation is based on a *conceptual structure* which we call DM/AM (for *domain model and adaptation model*) and on information about the user, stored in UM (for *user model*). In AHA! the adaptation consists of:

- Content adaptation: AHA! allows for the *conditional inclusion of objects*. Whenever an <object> tag is used in an XHTML page (or the <ref> tag in SMIL documents) with a "type" of aha/text, the DM/AM is consulted to select which actual resource (file) is included, depending on UM values. Conditionally included objects must be valid XML fragments, meaning that they must have a "root" tag. They need not be complete XHTML (or SMIL) files. The conditional inclusion of objects, introduced in AHA! 3.0, has an important advantage over conditional inclusion of fragments embedded in the page, as in older AHA! versions, namely that when the same additional or prerequisite explanation is needed in different places it need not exist multiple times. Also, conditionally included objects may themselves include other objects. This makes for a very powerful content adaptation mechanism.
- Link adaptation: AHA! performs *link annotation*, by using three link colors: *good* for recommended links, *neutral* for recommended links to previously visited pages, and *bad* for links that are not recommended. The default color scheme in AHA! is *blue, purple, black*, which results in *link hiding* (for the non-recommended links), but the color scheme can be changed either through a style sheet or through a form that lets end-users change the color scheme.

The author interacts with AHA! in two ways: a *conceptual* way, through authoring tools, and a *content-related* way, through any means the webserver offers for uploading data. We will concentrate on the authoring part that deals with authoring the *conceptual* structure of an AHA! application. Regarding the *content* an AHA! application consists of a directory tree, accessible through the website. As a result any facility the webserver offers for manipulating publicly visible data can be used to manipulate the content of an AHA! application. The AHA! website (http://aha.win.tue.nl/) contains an AHA! tutorial that includes the typical content elements.

- The starting point of every AHA! application is a login and/or registration page. This is an HTML page with a form used to register new users or to let existing users log in. Logins can be anonymous or can use a human-readable user identity (and name and password). The form also contains a number of hidden fields to identify the name of the application, the root of the directory tree and the name of the first page to show.
- When logged in the access to the pages is redirected through the AHA! "get" servlet. This servlet performs the content- and link adaptation. Apart from this the pages may look like a normal website. Links between pages are normal links, and so are references to images or applets to include.

The adaptation in AHA! is based on *concepts* and *adaptation rules*. When the end-user clicks on a link the AHA! engine consults a lookup-table to find the concept that corresponds to the requested *resource*. The adaptation rules associated with the *access* attribute of that concept are triggered. The rules are *event-condition-action* rules. When a rule is triggered a condition is checked. This is a Boolean expression using attributes of concepts. The attribute values are stored in UM (the user model). If the expression is true then the associated action is performed. Actions are updates to attribute values for concepts. Each time the attribute value of a concept is updated this triggers the event-condition-action rules associated with that attribute of that concept. Rules thus trigger each other, together performing perhaps complicated user model updates. The AHA! authoring tools warn an author for any danger of the rules potentially triggering each other indefinitely. In the next section we describe the authoring tools through which an author creates the concepts and adaptation rules, either directly (through the *concept editor*) or indirectly (through the *graph author*).

The AHA! Authoring Tools

AHA! comes with two main authoring tools. For fast and easy authoring of the conceptual structure there is the *graph author*, in which the author can use high-level concept relationships such as *prerequisites*. The tool generates the event-condition-action rules for the AHA! DM/AM automatically. For specialists or for fine-tuning the generated rules AHA! offers the *concept editor* (sometimes called *generatelist editor*). We first look at this low-level tool as it teaches us the "real" structure of the AHA! authoring formats and it helps us explain the more advanced aspects of the high-level graph author tool.

Figure 1a shows the main window of the Concept Editor. Every concept has a number of attributes (in this case “access”, “knowledge”, “visited”, “suitability” and “interest”). The “suitability” is shown as a *requirement* that is used to determine whether the concept is suitable or not.

For every attribute there are a number of properties and a number of event-condition-action rules. These are shown in Figure 1b. The properties include “Changeable” (meaning that the end-user is allowed to change the value of the attribute through a special form), “Persistent” (meaning that the value is permanently stored in the user model) and “System” (meaning that the attribute has a special meaning to the AHA! engine. This is true for the “access” attribute which triggers the rules and for the “visited” attribute which is used to decide between the *good* and *neutral* link colors. The adaptation rules are triggered by an update of the attribute value.

Figure 1c shows two additional aspects of an attribute: the “stability” property, which prevents a second adaptation when pages are revisited, and the conditional object inclusion, which we describe more in detail with the graph author.

Figures 1d and 1e show the event-condition-action rules in detail. Figure 1d is associated with the *access* event. Its condition says that the rule is only executed when the tutorial’s welcome concept’s suitability is false and when its knowledge is low (below 35). If this condition is met the knowledge of the welcome concept is set to 35. So this rule means that when visiting a non-recommended concept the knowledge can only increase, and will increase to 35.

The “Propagating” checkmark indicates that when the rule is executed it triggers the rules for the attributes (of concepts) that are updated in the rule’s action. In this case the tutorial.welcome.knowledge attribute is updated, so this triggers the rule shown in Figure 2e. That rule (not shown completely) updates the knowledge attribute of the tutorial concept. It adds 10% of the change to the welcome concept’s knowledge value. (The “_” symbol indicates *change* instead of *value*. This rule thus shows how knowledge is propagated up a concept hierarchy.

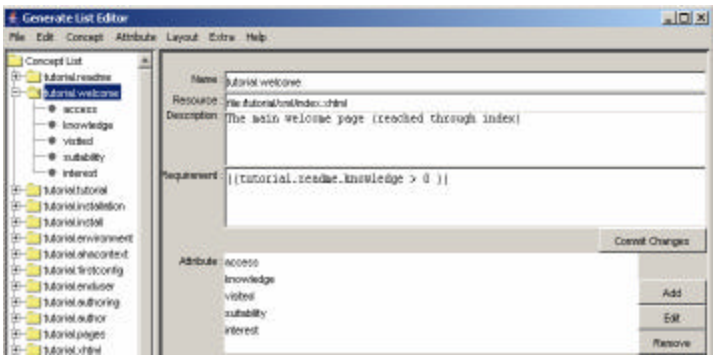

Figure 1a: Main Concept Editor window showing a concept.

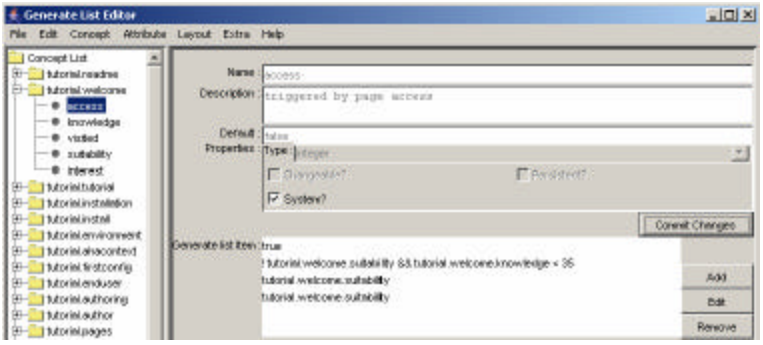

Figure 1b: Concept Editor view of a single attribute.

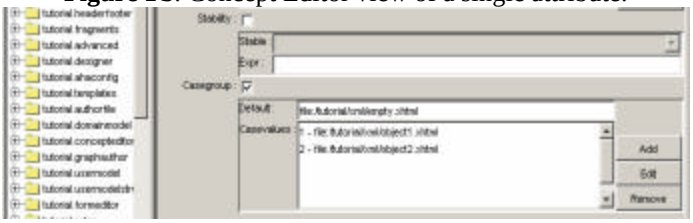

Figure 1c: Concept Editor view of conditional object inclusion.

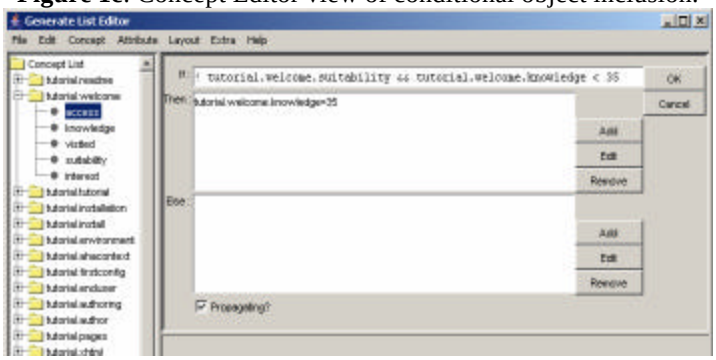

Figure 1d: Concept Editor view of event-condition-action rule.

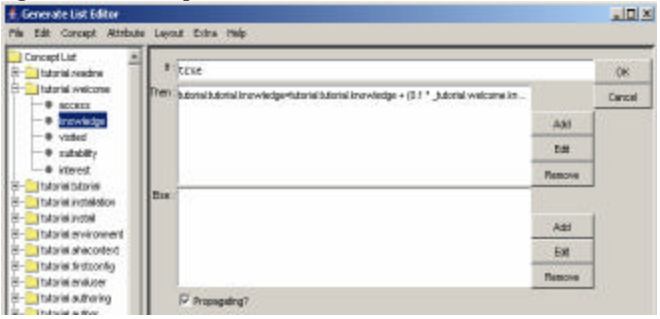

Figure 1e: Concept Editor view of another rule.

The concept editor allows for complete control over the user model updates associated with the user's browsing, and thereby also over the adaptation that is based on these updates, on the "requirement" associated with each concept and on the object inclusion. However, the example above also shows that defining an application's adaptation involves a lot of repetitive work. The rules for updating a concept's knowledge (depending on its suitability) are normally the same for all concepts. And the propagation of knowledge through a concept hierarchy is normally also done in a uniform way. Likewise, the *requirement* for a concept (determining its suitability) is also likely to depend on other concepts in a structured way, for implementing adaptation through *prerequisites*. There is thus a large part of the authoring work that can be automated. The *graph author* is a tool, first introduced in (De Bra et al, 2002) that lets authors "draw" the conceptual structure, and that generates the event-condition-action rules automatically.

There is no magical tool that can automatically structure a textbook for you, let alone generate the adaptation. A textbook normally consists of *topics* or *concepts*, that are grouped together to form a hierarchical structure, typically of *sections* and *chapters*. When the textbook is presented through hypertext (or a website) there is no restriction that every topic can belong to only one section or chapter. The navigational freedom a hypertext textbook gives to end-users is the main cause for learners to encounter difficulties. They can easily browse to an advanced topic before studying more basic definitions. By using prerequisites for adaptive link annotation or hiding users can be warned not to go to pages they are not ready to study. Alternatively, conditional object inclusion can be used to provide some additional explanations to compensate for missing knowledge.

Figure 2a shows the main graph author window. On the left we see the concept hierarchy. Knowledge propagation according to the concept hierarchy is generated automatically. On the right we see a graph of concept relationships. Concepts are simply dragged from the hierarchy frame to the graph. The relationships are created by selecting a relationship type (top right in the window) and then drawing an arrow from source to destination. By clicking on the arrow a parameter entry box appears. In the example we see red dashed arrows representing prerequisite relationships, and green solid arrows representing a "menu" relationship type we invented to make a menu-style navigation aid work in the tutorial. The menu arrows have a label representing the parameter (added as described above).

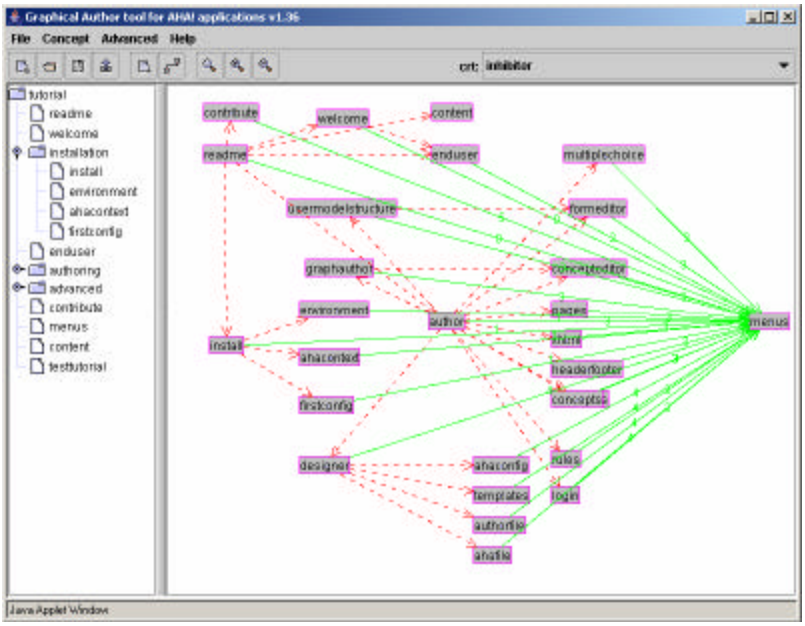

Figure 2a: Main Graph Author window.

When adding a new concept the dialog box shown in Figure 2b lets you select between different *templates*. These predefine which attributes the concept has, and whether a resource (page) must or can be associated with the concept. Other parameters are the same as with the concept editor.

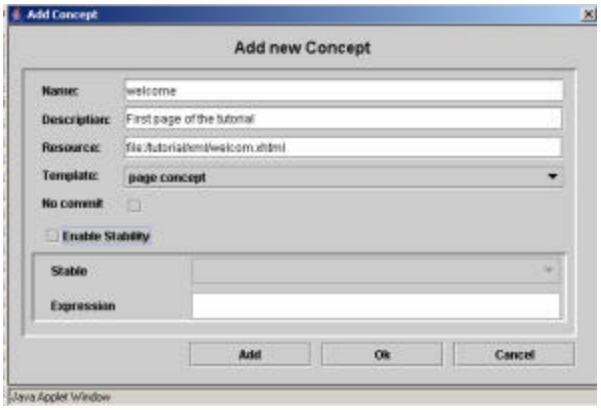

Figure 2b: Creating a new concept.

Figure 2c shows the conditional object inclusion dialog box. A conditionally included object always refers to a concept (not to a resource or file). The dialog box lets you enter an expression (just like the "requirement" or the "condition" of adaptation rules) that determines under which circumstances a resource is included. The first clause

that matches is included, so the conditions for different resources may overlap.

The graph author is a *generic* tool. Its functionality is largely determined by available *templates*: there are templates for concepts (defining the attributes and unary concept relationships like updating the knowledge upon access), and there are templates for concept relationships, defining how instances of the relationships are translated into the event-condition-action rules used by the AHA! engine. Because of lack of space we cannot describe the templates in detail in this paper.

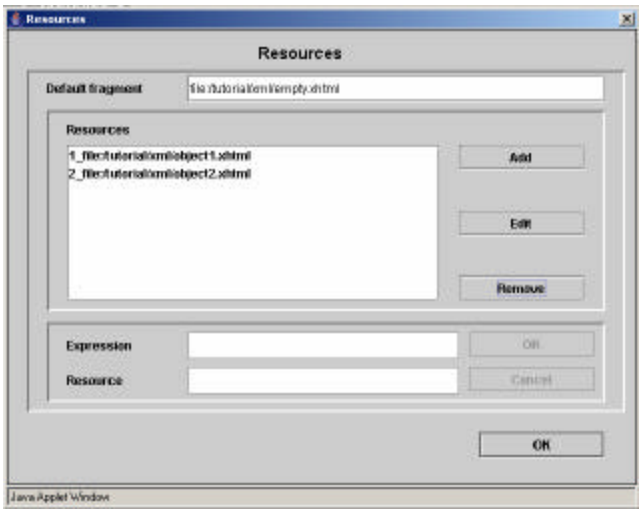

Figure 2c: Conditional object inclusion.

Additional Features and Tools

AHA! offers a few special function that are typically used in adaptive textbooks (and less in other applications). They can be called upon from a *header* or *footer* file.

Paul De Bra (debra@win.tue.nl) has read 27 pages and still has 1 page to read - list of read pages - pages still to be read
Changeable settings: link colors - knowledge of AHA! new tutorial - password - Log off

Figure 3: Header of the AHA! tutorial.

Figure 3 shows typical items of a header file. Information that is shown is generated through a <variable> tag, and links to special forms or reports through a <handler> tag.

- <username> generates the user's name (Paul De Bra in the example), and thus allows to give an AHA! application to have a "personal" flavor.
- <email> generates the user's email address.
- <numberdone> generates the number of pages the user has read, at least according to the *visited* attribute in the user model. (This may or may not correspond to reading pages, depending on the adaptation rules.)
- <numbertodo> generates the number of pages the user still has to read, again according to *visited*.
- <todo> generates the list of pages still to be read (according to *visited*).
- <done> generates the list of pages that have been read (according to *visited*).
- <colorsettings> lets the end-user change the *good-neutral-bad* color scheme through a form.
- <knowledgesettings> lets the end-user change the *knowledge value* for concepts through a form. Only concept with a "changeable" knowledge attribute are included in this form.
- <passwordsettings> presents a form for changing the user's password. (For anonymous users a password cannot be entered.)
- <logoff> ends the session, thus preventing others from continuing a session if the browser is left active when the user leaves the computer.

Apart from these standard features AHA! offers two special kinds of forms: *multiple-choice tests* and *custom forms*. Multiple-choice tests in AHA! consist of a number of questions that can have two forms: with a single and with multiple correct answers. You can provide more answers than are presented to the learner. The learner can get just a score or a more elaborate feedback. An authoring tool for multiple-choice tests has recently been developed by Cristobal Romero from the University of Cordoba (Spain), and will be added to the AHA! distribution.

AHA! also offers the possibility for creating custom forms. These let you offer the end-user the possibility to change values for arbitrary (changeable) attributes of concepts.

The form editor, shown in Figure 4, consists of a standard HTML editor frame, coupled with a user interface for adding form elements that are tied to attributes of concepts. Forms, created with the form editor, are useful for entering preferences, interests, background knowledge, etc. Some care has to be taken when using the form editor that the generated file is acceptable for an XML parser. In the current version of the J2SE that is used the included editor is not yet fully XHTML compliant.

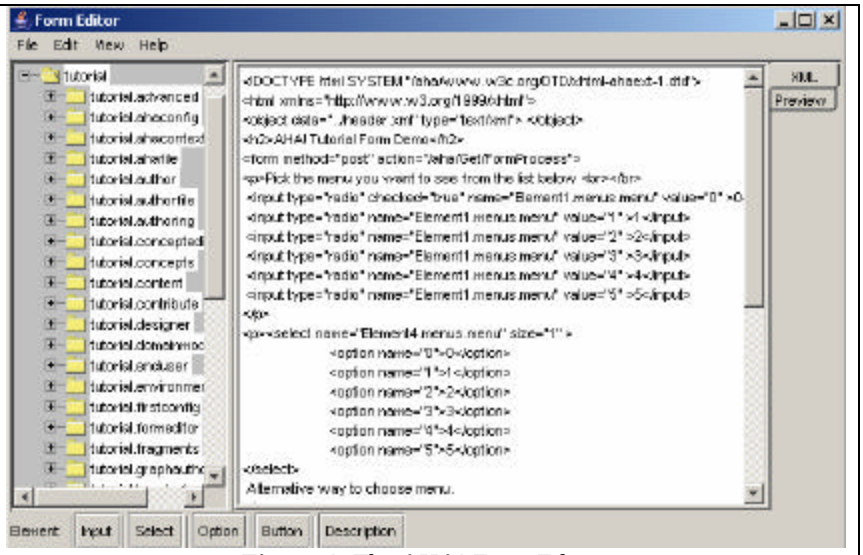

Figure 4: The AHA! Form Editor.

Conclusions

In this paper we have provided an overview of AHA!, thereby concentrating on the authoring tools that are available for creating the conceptual structure of an adaptive textbook. In the interactive roundtable session we will walk through the creation of an adaptive textbook (for which the content pre-exists). The aim is to not only show the authoring process but also to answer questions and to gather ideas for new authoring tools.

References

Brusilovsky, P. (1996). *Methods and Techniques of Adaptive Hypermedia*. User Modeling and User-Adapted Interaction, 6, (pp. 87-129). (Reprinted in Adaptive Hypertext and Hypermedia, Kluwer Academic Publishers, 1998, pp. 1-43.)

Brusilovsky, P. (2001). *Adaptive hypermedia*. User Modeling and User Adapted Interaction, 11 (1/2) pp. 87-110.

Brusilovsky, P., Eklund, J., Schwarz, E. (1998). *Web-based Education for All: A Tool for Developing Adaptive Courseware*. Computer Networks and ISDN Systems (Seventh International World Wide Web Conference), 30, 1-7, 291-300.

Cini, A., Valdeni de Lima, J. (2002). *Adaptivity Conditions Evaluation for the User of Hypermedia Presentations Built with AHA!.* Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, Springer Verlag, LNCS 2347, pp. 490-493.

Cristea, A., de Mooij, A. (2003). *Adaptive Course Authoring: My Online Teacher*. ICT'03 (International Conference on Telecommunications), Papeete, French Polynesia, IEEE,IEE, ISBN: 0-7803-7662-5, pp. 1762-1769.

De Bra, P., Aerts, A., Rousseau, B. (2002). *Concept Relationship Types for AHA! 2.0*. Proceedings of the AACE ELearn'2002 conference, October 2002, pp. 1386-1389.

De Bra, P., Aerts, A., Berden, B., De Lange, B. (2003a) *Escape from the Tyranny of the Textbook: Adaptive Object Inclusion in AHA!*. Proceedings of the AACE ELearn 2003 Conference, Phoenix, Arizona, November 2003, pp. 65-71.

De Bra, P., Santic, T., Brusilovsky, P. (2003b). *AHA! meets Interbook, and more...* Proceedings of the AACE ELearn 2003 Conference, Phoenix, Arizona, November 2003, pp. 57-64.

Romero, C., De Bra, P., Ventura, S., de Castro, C. (2002). *Using Knowledge Levels with AHA! for Discovering Interesting Relationships.* Proceedings of the AACE ELearn'2002 Conference.

Romero, C., Ventura, S., De Bra, P., De Castro, C. (2003). *Discovering Prediction Rules in AHA! Courses*. Proceedings of the User Modeling Conference, Johnstown, Pennsylvania, June 2003, pp. 35-44.