Escape from the Tyranny of the Textbook: Adaptive Object Inclusion in AHA!

Paul De Bra[*], Ad Aerts, Bart Berden, Barend de Lange
Department of Computing Science
Eindhoven University of Technology (TUE)
PO Box 513, Eindhoven, The Netherlands
debra@win.tue.nl

Abstract: 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 paper we present a new and more flexible way to conditionally include information: *adaptive object inclusion*. This technique breaks the one-to-one connection between fragments and the pages on which they appear, and thus makes the adaptive textbook less textbook-like. We realize that when applications become more and more dynamic users may start to feel uneasy when the presentation of the same information or concepts changes all the time (because the system adapts to a changing user model). We therefore also introduce the idea of *stable presentations*, a technique to limit the adaptation to information the user has not yet visited. We have implemented both new techniques in the AHA! system. At the same time we have made the document format completely independent of AHA!. This makes it possible to create textbooks using standard tools for (X)HTML or for other XML formats like SMIL.

Introduction and Background

Adaptive hypermedia has emerged as a research field around 1990. Researchers from areas like adaptive, user-model based user interfaces, intelligent tutoring systems and hypermedia systems started converging in the direction of offering access to information through interfaces that are adaptive based on user modeling, that are less restrictive (to the user) than the ITS (of that time), and that use hypermedia as their interface paradigm. Brusilovsky (Brusilovsky, 1996) has given an excellent overview of adaptive hypermedia methods, techniques and systems. The overview was updated later (Brusilovsky, 2001) to include newer developments. Although adaptive hypermedia systems have been and are being used in different application areas, the area of education is (still) very dominant. Most effort seems to have gone into adaptive “manipulation” of the links in a textbook. The new overview (Brusilovsky, 2001) mentions *direct guidance*, *adaptive link sorting*, *adaptive link hiding* (with three variants *hiding*, *disabling* and *removal*), *adaptive link annotation*, *adaptive link generation* and *map adaptation*. The overall aim of these methods and techniques is to help the user in selecting “appropriate” links, i.e. links that lead to information the user is interested in, ready to learn, etc. The choice for the emphasis on link adaptation seems logical. In a classical (paper) textbook the author can assume that all learners read/study the book in the same order (from begin to end). The use of hypermedia gives the learner the possibility to choose his or her own reading order, by selecting one of the many links that are available on each page. Not all possible reading orders that result from these choices are “wise” choices. Link adaptation helps users in selecting “appropriate” links. Well-known systems like Interbook (Brusilovsky et al, 1998) and KBS Hyperbook (Henze & Nejdl, 1999) focus on guiding the learner through adaptively generated or annotated links.

There is an alternative way to help users in coping with the navigational freedom: the information that is shown on a page, or the presentation thereof, can be adapted to the user. So instead of, or in addition to, deciding whether or not a page is “suitable” for the user, the system can opt to *make* the page suitable by changing what is shown to the user when he or she requests that page. Brusilovsky (Brusilovsky, 2001) mentions *inserting/removing fragments*, *altering fragments*, *stretchtext*, *sorting fragments* and *dimming fragments* as possibilities for altering what is shown to the user. All these techniques seem to suggest that a page consists of a set of information items that

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

are manipulated in some way when the user requests the page. In AHA! (De Bra & Calvi, 1998, De Bra et al, 2002a) the conditional inclusion of fragments (called *inserting/removing fragments* in Brusilovsky's overview) is used to conditionally show some additional (prerequisite) explanations when needed. Our experience with this technique (in the hypermedia course http://wwwis.win.tue.nl/2L690/) shows that short explanations that are added and that are not shown later on when a user revisits a page are not missed (or even noticed) by the learner afterwards. However, it is to be expected that extensive use of the conditional inclusion of fragments could potentially disturb users. Several researchers have already experimented with AHA! for different purposes (Cini et al, 2002, Romero et al, 2002), and have not found such problems. On the other hand, the experiments did not have the purpose to discover such difficulties, but concentrated on measuring adaptivity (Cini et al, 2002) and on the use of data mining on the logging information in order to improve adaptive courses (Romero et al, 2002).

In this paper we describe two additions to the AHA! system that increase the versatility of the tool. We first describe an extension of conditional inclusion of fragments, whereby the "fragments" become external objects. This has the advantage that when a fragment (explanation) is sometimes needed in different places (on different pages) it can be stored just once and conditionally included where and whenever needed. This effectively breaks the "page" notion of the textbook: an adaptive course presents concepts, and includes objects that are needed to explain these concepts. To compensate for the potentially increased use of conditional inclusion of information in many places we describe a second extension of the AHA! system: the ability to generate *stable presentations*. In a stable presentation the adaptation of an object is performed just once, when the user first accesses the object. Whenever the object is revisited (on the same or on another page) it is presented as before. Our extension actually has more options for stability. An important reason for using *stable presentations* is to prevent explanations from disappearing. When the system decides that an explanation (object) is no longer needed it will still be shown as long as the stability requirement defines. (This can be until a certain condition is met, until the end of a session, or even forever.)

Object inclusion in AHA!

In AHA! version 2.0 (De Bra et al, 2002a) the versatility of AHA! was greatly increased through a richer domain, adaptation and user model. Tools were developed to make authoring of the conceptual structure (domain model) and the adaptation easier. But writing the pages of a course text still remained somewhat problematic because AHA! 2.0 uses XHTML with a small extension. The extension consists of `<if>` and `<block>` tags that are used for the conditional inclusion of fragments. For instance, a page from a cookbook may include a conditional fragment like:

```
<if expr="ahacook.bainmarie.knowledge==0">
   <block>
      The technique "au bain marie", or "baking with a water bath",
      consists of placing a container of food in a large, shallow pan
      filled with warm water. The water surrounds and protects delicate...
   </block>
</if>
```

The fragment is shown if the user has no knowledge of the concept "bainmarie" in the course "ahacook". While seemingly easy to write, the use of `<if>` and `<block>` tags prevent authors from using validating (X)HTML editors that do not allow the use of tags that do not exist in (X)HTML.

The first purpose of the introduction of conditional object inclusion was to eliminate the non-standard tags in pages. A secondary goal was to also eliminate the need to know the exact syntax of the expressions that are used in the conditions. A final goal was to allot the inclusion of objects in different pages without redundant storage.

Design choices for object inclusion

Several options were investigated to eliminate the use of non-standard XHTML, and at the same time eliminate the inclusion of the fragment text in the page. When a fragment is stored as a separate file it can be included in many pages without duplicating the text. Options to indicate the inclusion of a fragment in a page are:

- IFrames were the first technique we investigated to include fragments. They are part of HTML and can thus created using HTML editors. Furthermore, there normal function is already a type of object inclusion. We opted against the use of IFrames because their use in AHA! would make it impossible to create presentations that already contain IFrames that should not be intercepted by AHA!. Another problem with IFrames is that they have a fixed size, whereas the fragments to be included may have different sizes. (This is not a serious problem as the AHA! engine can change the size parameters.)

- Server-Side Includes (SSI) normally use a server-side extension to include files into a page. They use a structured HTML comment (`<!--#INCLUDE FILE="filename.shtml"-->`). We have opted against this mechanism because it is not based on a real tag. A previous AHA! version also used HTML comments. We found that creating such comments was made difficult or impossible in some editors.
- Objects, using the `<object>` tag. In (X)HTML the object tag is supposed to become the replacement for many other tags that are currently used to include external objects, including "img", "applet" and "embed". Unlike with IFrames an object is given a type, to tell the browser how to present the object (as an image, applet, or through some other special viewer). The following construct can be used to tell the AHA! servlet that an object is a conditional fragment:
```
<object name="fragment-name" type="aha/text">
   alternative text displayed if object inclusion fails
</object>
```
  When the page is viewed without using the AHA! engine the type "aha/text" is unknown and the browser will show the alternative text. When used with AHA! the engine will examine the adaptation and user models to decide whether to include the object or not and will choose between actual files to include.

When the AHA! engine encounters an object tag it will only process it when the type is "aha/text". Other object tags are passed to the browser as is. The intended use of object tags in HTML is thus not disturbed by our use of the same tag for conditional inclusion of fragments.

Implementation of conditionally included objects in AHA!

Each time a user requests a page, to be served by the AHA! engine, the engine performs the following actions in sequence:

1. The user model is retrieved (if not already in memory).
2. The Boolean *access* attribute of the concept corresponding to the requested page is temporarily set to true.
3. The adaptation rules are executed, as explained in previous papers, e.g. (De Bra et al, 2002a), resulting in user model updates. For instance, when a page is accessed the *knowledge* value for the corresponding concept is increased. Also, this knowledge increase is (partially) propagated to some higher level concepts like a section, chapter or whole textbook.
4. The page is sent to the user's browser.

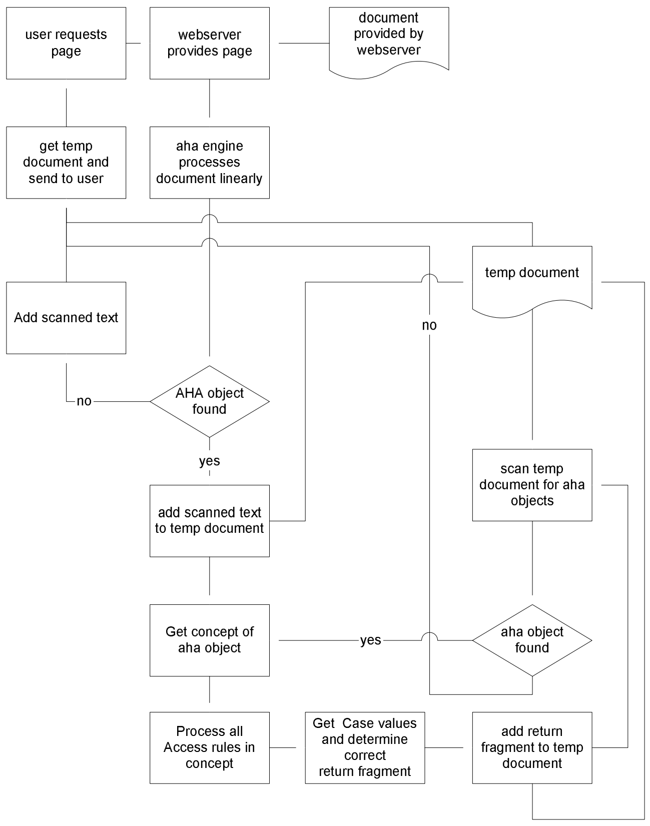
When a page contains conditionally included objects, what happens now is that while the engine is parsing the page in step 4, each time an object is encountered the adaptation engine is started again, with the *access* event for the concept that corresponds to the object. The engine may decide upon the inclusion of some XHTML file fragment. This fragment is not simply sent to the browser, but is parsed as well. It too may include conditionally included objects. The engine will ensure that all recursively included objects are retrieved and if necessary included. After the inclusion of the object (and its recursive descendants) the parsing of the page continues. (Fig. 1) shows the whole process in a graphical way.

In order for the adaptation engine to decide how to update the user model when an object is accessed and how to decide which file to actually include for the object, the author has to write adaptation rules that are essentially the same as for concepts that correspond to pages, or for high-level abstract concepts. This format was explained in (De Bra et al, 2002a). We give a small example below, in a shortened simplified syntax, for the conditional inclusion of an explanation of a concept *au bain marie* in a cookbook.

```
<concept>
   <name>ahacook.au_bain_marie</name>
   <expr>ahacook.au_bain_marie.knowledge!=100</expr>
   <attribute>
      <name>access</name>
      <action>
         <if>ahacook.au_bain_marie.knowledge<50</if>
         <then>ahacook.au_bain_marie.showability := 0</then>
      </action>
      <action>
         <if>ahacook.au_bain_marie.knowledge>=50</if>
         <then>ahacook.au_bain_marie.showability := 50</then>
      </action>
      <action>
         <if>ahacook.au_bain_marie.knowledge==100</if>
         <then>ahacook.au_bain_marie.showability:=100</then>
      </action>
      <action>
```

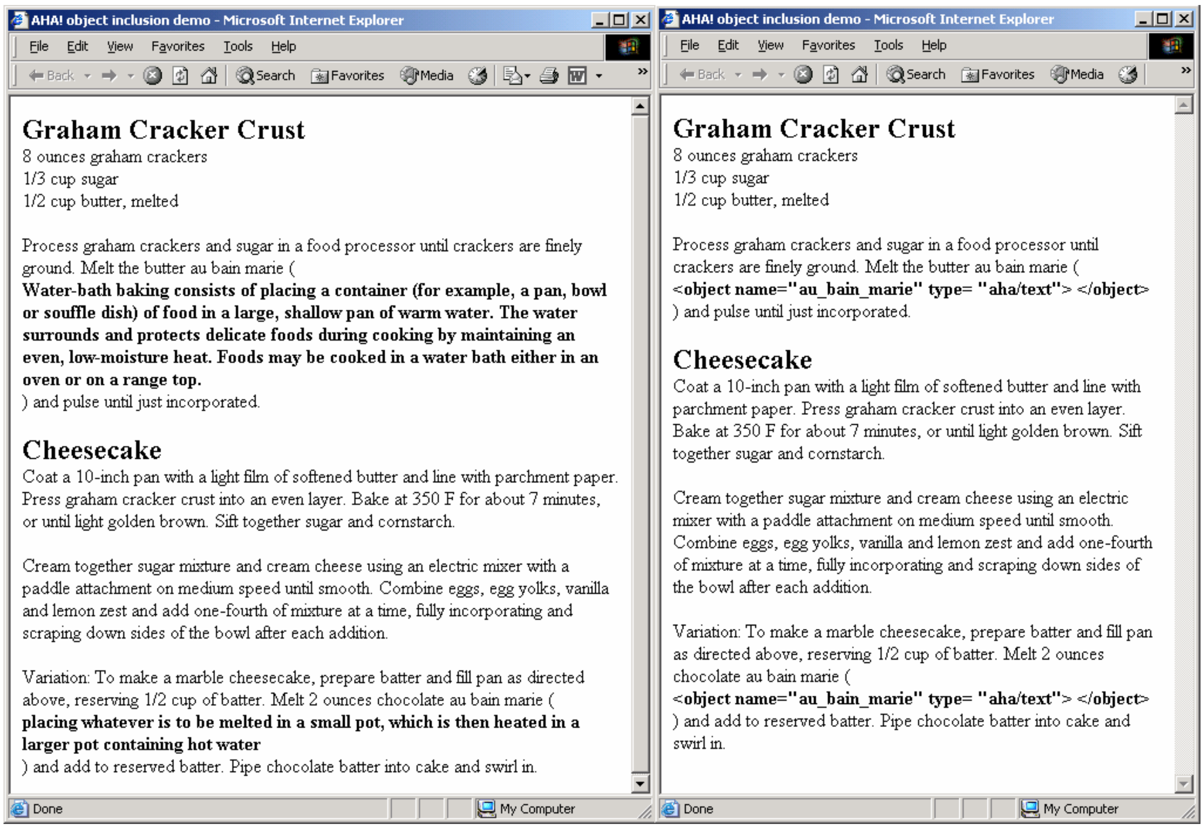
```
            <if>ahacook.au_au_bain_marie.knowledge&lt;100</if>
            <then>ahacook.au_bain_marie.knowledge += 50</then>
         </action>
      </attribute>
      ...
      <attribute>
      <attribute>
         <name>showability</name>
         <casegroup>
            <defaultfragment>au_bain_marie.xhtml</defaultfragment>
            <casevalue>
               <value>0</value>
               <returnfragment>frag_au_bain_marie_large.xhtml</returnfragment>
            </casevalue>
            <casevalue>
               <value>50</value>
               <returnfragment>frag_au_bain_marie_medium.xhtml</returnfragment>
            </casevalue>
            <casevalue>
               <value>100</value>
               <returnfragment>frag_au_bain_marie_small.xhtml</returnfragment>
            </casevalue>
         </casegroup>
      </attribute>
   </concept>
```

This (simplified) example shows that accessing the “au bain marie” object first of all updates the knowledge value for this concept. The knowledge value then influences the “showability” attribute, which in turn determines the selection of the “returnfragment”. Authors need not know the given syntax as they use a tool like the “concept editor” described in (De Bra et al, 2002a) or the “graph author” described in (De Bra et al, 2002b) to define the concept structure and adaptation rules. Both tools have been updated to accommodate the new object inclusion.



user requests page
webserver provides page
document provided by webserver
get temp document and send to user
aha engine processes document linearly
temp document
Add scanned text
no
AHA object found
no
yes
add scanned text to temp document
scan temp document for aha objects
Get concept of aha object
yes
aha object found
Process all Access rules in concept
Get Case values and determine correct return fragment
add return fragment to temp document

(Fig. 2) shows a page from a (fictitious) cookbook. It contains two recipes that use the au bain marie cooking technique. With the first recipe an explanation of this technique is included. This inclusion triggers a user model update, registering knowledge of this concept. With the second recipe the explanation of au bain marie is also included. But because the knowledge value of the concept has increased a shorter explanation is shown.


AHA! object inclusion demo - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Done
My Computer

Graham Cracker Crust
8 ounces graham crackers
1/3 cup sugar
1/2 cup butter, melted

Process graham crackers and sugar in a food processor until crackers are finely ground. Melt the butter au bain marie (
Water-bath baking consists of placing a container (for example, a pan, bowl or souffle dish) of food in a large, shallow pan of warm water. The water surrounds and protects delicate foods during cooking by maintaining an even, low-moisture heat. Foods may be cooked in a water bath either in an oven or on a range top.
) and pulse until just incorporated.

Cheesecake
Coat a 10-inch pan with a light film of softened butter and line with parchment paper. Press graham cracker crust into an even layer. Bake at 350 F for about 7 minutes, or until light golden brown. Sift together sugar and cornstarch.

Cream together sugar mixture and cream cheese using an electric mixer with a paddle attachment on medium speed until smooth. Combine eggs, egg yolks, vanilla and lemon zest and add one-fourth of mixture at a time, fully incorporating and scraping down sides of the bowl after each addition.

Variation: To make a marble cheesecake, prepare batter and fill pan as directed above, reserving 1/2 cup of batter. Melt 2 ounces chocolate au bain marie (
placing whatever is to be melted in a small pot, which is then heated in a larger pot containing hot water
) and add to reserved batter. Pipe chocolate batter into cake and swirl in.

Graham Cracker Crust
8 ounces graham crackers
1/3 cup sugar
1/2 cup butter, melted

Process graham crackers and sugar in a food processor until crackers are finely ground. Melt the butter au bain marie (
<object name="au_bain_marie" type= "aha/text"> </object>
) and pulse until just incorporated.

Cheesecake
Coat a 10-inch pan with a light film of softened butter and line with parchment paper. Press graham cracker crust into an even layer. Bake at 350 F for about 7 minutes, or until light golden brown. Sift together sugar and cornstarch.

Cream together sugar mixture and cream cheese using an electric mixer with a paddle attachment on medium speed until smooth. Combine eggs, egg yolks, vanilla and lemon zest and add one-fourth of mixture at a time, fully incorporating and scraping down sides of the bowl after each addition.

Variation: To make a marble cheesecake, prepare batter and fill pan as directed above, reserving 1/2 cup of batter. Melt 2 ounces chocolate au bain marie (
<object name="au_bain_marie" type= "aha/text"> </object>
) and add to reserved batter. Pipe chocolate batter into cake and swirl in.

Figure 2: Left: page from cookbook with two “au bain marie” examples,
Right: the corresponding object tags in the page

Stable Presentations

Normally, in AHA! as in most other adaptive hypermedia systems, each page a user requests is adapted to the user model at the time of the request. We will first examine exactly what this *at the time of the request* means in the new version of AHA! with conditional inclusion of objects. If we look back at the process followed by the AHA! engine we see that *first* the user model is updated and *second* the page is adapted based on the *new* state of the user model. This is also what other systems do for their link adaptation. Once a page is presented to the user it cannot be changed any more without an explicit user interaction. So the decision which links to recommend to the user and which links to hide or warn against must be based on the user’s knowledge state *after reading the page*.

For the inclusion of objects the situation is the other way around. In order to decide whether an explanation needs to be shown the knowledge state of the user needs to be considered *before reading the page*. Otherwise the explanation would never be included. Note that in the “bain marie” example we gave before the “showability” state of the object was determined *before* updating the knowledge value, exactly for this reason.

Especially for users with a visual memory it is important that once a page is presented in some way (with a certain choice of objects) the presentation stays that way when the user revisits that page. In AHA! we have added three options for keeping the presentation of an object (like an explanation) stable:

1. *stable forever*: Once a user has accessed an object and it has been decided how to present that object (i.e. which file to use for the object) the presentation stays the same forever (for this user). If this stability

setting is chosen for all objects the user will never notice any adaptation. The course text or website will appear to be static, but differences can be seen when comparing the presentation for different users. (Of course link adaptation still goes on to guide the user through the course material.)

2. *session-stable*: The presentation of the object remains stable during a browsing session, but when the user logs out and later returns to the course, adaptation resumes. This is especially useful for adaptation in websites that are not consulted very frequently. (Subsequent sessions do then not exist because of a browser or operating system crash.) When returning to a website after days or weeks it is perhaps best to have the site adapt to the user model, and not retain some presentation settings from an old session.
3. *expression-stable*: The presentation of the object remains stable as long as some (user model) expression remains true. An example of an application of this technique is to present the same (version of an) explanation until a threshold for the knowledge of a concept is reached. The basic idea is that small changes in the user model would not cause visible adaptation but large changes would.

Currently the stability is tied to the objects. A stable object is presented in the same way wherever it appears. This is not always convenient. In the “bain marie” example of (Fig. 2) for instance, a stable explanation of the concept would imply that the same explanation would appear in both places. We are considering adding stability of objects in combination with the page they appear on, and with their position on that page. We are currently investigating ways to achieve more “stability” options without incurring too much storage overhead in the user model.

Conclusions and Future Work

Adaptive hypermedia used in education has focused on adaptive electronic textbooks, as evidenced by existing systems and courses (De Bra & Calvi, 1998, Brusilovsky et al, 1998, Henze & Nejdl, 1999). With the latest additions to the AHA! system we have largely severed the static link between pages and the page content. Adaptation rules can now be used to decide which objects to present on which pages, based on the user model. This *conditional inclusion of objects* enables the creation of less textbook-like adaptive courses. A completely different potential application of the object inclusion extension is its use to select between different presentation styles. Concepts can be illustrated with images, audio fragments, video, or additional textual explanations. Adaptation can be used to select between such presentation formats because images, audio fragments, video and text fragments are typically stored in separate files, and can be easily included as objects.

The addition of *stable presentations* in AHA! was introduced, mainly as a tool for diminishing the potential problems of users with too much content adaptation. As with previous versions of AHA!, we have provided a versatile tool, a framework, not an educational application. We will be developing new course material but like with previous AHA! versions most usable feedback and usability evaluation needs to come from others who use AHA! for research or education (or other) purposes. Only experience will tell whether the new tools for more adaptation on the one hand (with object inclusion) and for less adaptation on the other hand (stable presentations) together form a valuable addition to our general-purpose adaptation platform.

Together with the extensions to the AHA! engine the authoring tools described in (De Bra et al, 2002a) and (De Bra et al, 2002b) have been extended to allow authors to use the new functionality with the existing type of authoring tools. We are considering the addition of completely new ways of authoring for AHA!. One future development is the definition of a kind of “programming language” that describes adaptation by describing the possible navigation paths in a program. Another development is that we are creating compilers from existing adaptive hypermedia systems to the AHA! system. One such compiler, from Interbook to AHA! is described in another ELearn’2003 submission.

Acknowledgements

The development of AHA! was made possible through a grant of the NLnet Foundation and the hospitality of the Eindhoven University of Technology hosting the NLnet-supported researchers who developed (and are further developing) the AHA! system and applications based on AHA!. We would also like to thank all the people who are using AHA! and providing us with the motivation and inspiration to continue to develop it. We especially thank Luca Botturi (Italy), Allesandra Cini (Brazil), Denghui (China), Carla Lawless (Ireland) and Cristobal Romero (Spain) for developing and showing different applications with or for the AHA! system.

References

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.

De Bra, P. & Calvi, L. (1998). *AHA: a Generic Adaptive Hypermedia System*. 2nd Workshop on Adaptive Hypertext and Hypermedia, pp. 1-10. (URL: http://wwwis.win.tue.nl/ah98/DeBra.html)

De Bra, P., Aerts, A., Smits, D., Stash, N. (2002a) *AHA! Version 2.0, More Adaptation Flexibility for Authors*. Proceedings of the AACE ELearn'2002 conference, October 2002, pp. 240-246.

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

De Bra, P., Houben, G.J. & Wu, H. (1999). *AHAM, A Dexter-based Reference Model for Adaptive Hypermedia*, Proceedings of the ACM Conference on Hypertext and Hypermedia, pp. 147-156.

Henze, N., Nejdl, W. (1999). *Adaptivity in the KBS Hyperbook System*. Second Workshop on Adaptive Systems and User Modeling on the World Wide Web, pp. 67-74. (URL: http://wwwis.win.tue.nl/asum99/henze/henze.html)

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.