

# Creating and Delivering Adaptive Courses with AHA!

Paul De Bra, David Smits, and Natalia Stash

Faculty of Mathematics and Computer Science, Eindhoven University of Technology,  
Postbus 513, 5600 MB Eindhoven, The Netherlands  
{debra, dsmits, nstash}@win.tue.nl

**Abstract.** AHA! is an Open Source adaptive hypermedia platform, resulting from 10 years of experience with creating, using and improving on-line adaptive courses and presentations. This paper focuses on some recent additions to AHA! that are especially important for adaptive educational applications, namely *stable presentations*, *adaptive link (icon) annotations* and *adaptive link destinations*. We not only describe the technical aspects of these parts of AHA! but also illustrate their use in educational applications. We describe some fundamental limitations of Web-based adaptive applications, and show how AHA! deals with them in order to provide adaptation to *prerequisite relationships* in the way one would expect.

## 1 Introduction and Motivation

World Wide Web has changed the world in more than one way. It has radically changed the way in which all kinds of organizations provide information to their employees, customers and the general public. It has also made it possible to have direct interaction between an end-user and provider. But it also changed the *form* in which information is available: linear documents (or hierarchically structured ones) have been replaced by *hypertext*. The navigational freedom that comes with hypertext has stimulated a *browsing* learning style, where the user looking for information or learning material simply picks out the items she is interested in (judging this based either on link anchors or titles or on the opinion of a search engine). This seemingly random user behavior (most suitable for *field-independent* [6] learners) makes it difficult for an author or a course designer to develop content and a link structure that can be fully understood, no matter which path the end-user decides to follow.

Adaptive hypermedia [2,3] comes to the rescue. In “normal” (or static) hypertext it is impossible to offer the right content, the needed definitions and explanations, and at the same time hide or deemphasize already known definitions and unnecessary explanations. When creating the content you have no way of knowing what the end-user has studied before jumping to a certain (web)page. With adaptive hypermedia it is possible to select and present information content based on the user’s previous actions (processed and stored in a *user model*) and to select and annotate links in a way that guides the user towards the most relevant information. As we will show in this paper, this added service for the end-user can be created and delivered with little additional effort on the author’s side.

At the TU/e we started offering a course on the topic of hypermedia in 1994 [8]. This course was offered through the Web, in hypermedia (actually mostly hypertext) form. Initially there were also lectures, but as of 1996 the course was reduced to just its on-line hypermedia form and made available to students of different Dutch and Belgian universities. In order to make it possible to fully utilize the hypertext nature of the course text, with a rich link structure and no hierarchical menu system that would constrain the learner's navigational freedom, we decided to make the course text adaptive. This development has resulted in the Adaptive Hypermedia Architecture AHA!, which is now available in its third major release. (See [9] and <http://aha.win.tue.nl/> for details about the AHA! project.) Feedback from authors, researchers and end-users has led us to create a rich feature set, with items that target a large spectrum of possible applications, and with items that are specifically useful for educational applications, including progress reports, inspection (and update) of the knowledge of all the concepts of a course, forms for setting preferences or altering other parts of the user model, and an (externally contributed) authoring and delivery tool for multiple-choice tests. Based on user experience we also designed the main authoring tool for AHA!: the *Graph Author* [10], used to define *concepts*, *concept relationships* and the (one-to-one or one-to-many) connection between concepts and *resources* (like webpages).

Fairly complete information on AHA! can be found on the project website [aha.win.tue.nl](http://aha.win.tue.nl), including a tutorial. In this paper we only focus on three novel aspects in AHA! that are especially useful for educational applications:

1. *stable presentations*: When an adaptive system continuously adapts the information it presents to the user model (or the user's estimated knowledge state) pages are adapted (and thus changed) each time the user visits them. We have grown accustomed to changing link colors (blue link anchors turning purple when they refer to already visited pages) but not to changing content. We expect a page to present the same information when we revisit it later (although some content is expected to be dynamic, like a weather or traffic report). The *stable presentations* technique in AHA! lets authors decide which pages or objects should always be adapted, or should only be adapted under certain conditions. (And by offering the learner a form to alter a value that influences the condition the end-user can also influence the decision to keep the presentation stable or not.)
2. *adaptive link (icon) annotations*: In several existing adaptive hypermedia systems the "status" of a link (recommended or not, interesting, already visited, etc.) is not or not only indicated through a change in the *color* of the link anchor, but also through some icon, like a small or large checkmark, or a colored ball (in ELM-ART [5], Interbook [4], NetCoach [17], KBS-Hyperbook [12]), etc. The balls or other icons can be placed in front of the link anchor or behind it. Each system has its own rules for deciding which icon to use under which circumstances. In AHA! an author can easily define these rules or conditions, for any arbitrary number of icons. Also, different *views* (or html frames) can use different icons with different conditions for selecting them.
3. *adaptive link destinations*: Link adaptation is not always a simple case of recommended vs. not recommended, but sometimes a matter of *where* to send the user to. A link to a topic may wish to refer a (*field-dependent* [6]) beginner to an introductory page on that topic, and an expert to an advanced page. In AHA! the

binding between the *concept* a link points to and the actual *resource* (or page) that is returned is dynamic. The link destination is selected adaptively, based on the user model.

In each of the following sections we highlight one of these adaptation techniques. We explain how to make use of them in AHA! and illustrate their use in an (educational) example. But first we briefly explain how most Web-based adaptive systems work, and how they deal with the fundamental limitations of using page-based HTTP requests and responses.

## 2 Architecture and Limitations of Web-Based AHS

AHA! is a typical Java-based web application, using Servlets in combination with a Java-based server like Tomcat. Figure 1 shows the global architecture of such a system, and Figure 2 shows the different files or databases used by the AHA! engine.

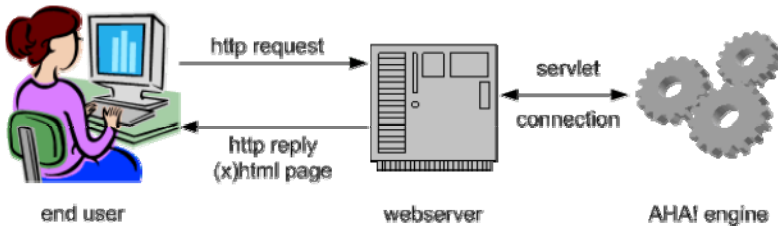


Fig. 1. Global architecture of a Web-based application

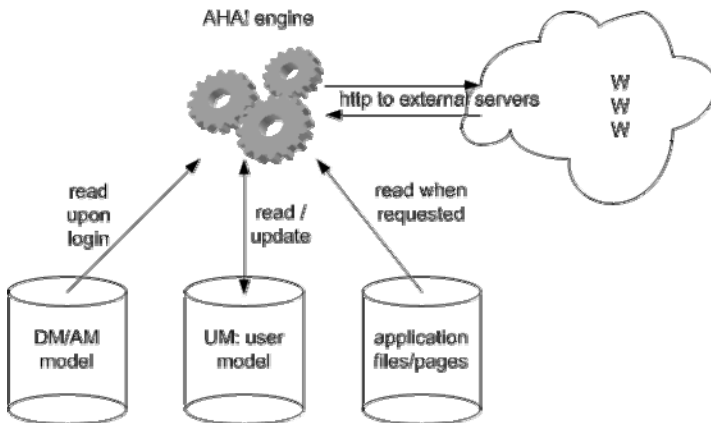


Fig. 2. Data stores used by the AHA! engine

In Figure 2 we see that the AHA! engine uses three types of local information, and potentially also pages that reside on other servers on the Web. The combined domain and adaptation model DM/AM represents a model of the conceptual structure of the application, and the adaptation rules. This information is retrieved when the end-user

logs in. As the user is interacting with the application a user model (UM) is used and constantly updated. The interaction involves (mostly) the retrieval of pages and other application files (like images), but it can also involve using forms to update UM or taking multiple-choice tests. DM/AM and UM are used to decide which application file or “page” to retrieve upon a request from the user (and which objects to conditionally include in that page). The request may also result in files being retrieved from other web servers.

As a “pure” Web-Based system, AHA! only reacts to HTTP requests and sends HTTP responses that contain a document (typically in HTML or XHTML, but any XML format is allowed). The interaction between the user and the AHA! application thus happens on a page by page basis. (AHA! allows the use of the novel “Ajax” technologies to allow requests for parts of a page that are updated without refreshing the whole page, but we have not yet considered the UM updates that one might wish to associate with such within-page interaction.) Each time the user issues an HTTP request (typically by clicking on a link) the following procedure is followed:

1. The request contains a URL that can refer to a concept (from DM) or to a page. In DM concepts belong to a hierarchy and can have an associated resource which is a page. When a request refers to a page, the corresponding concept is looked up. The procedure continues with the found concept. If a concept does not have an associated page AHA! will generate a menu-like page that links to sub-concepts from the concept hierarchy. (We will not consider this case in the sequel.)
2. UM is retrieved (if not already cached). AHA! normally caches UM during a session, but it updates the stored version after every request so that other applications can see the changes.
3. The AHA! engine executes rules from AM, starting from the rules associated with the “access” attribute of the requested concept/page. These rules cause updates to some attributes (attribute values) of some (UM) concepts.
4. The updates caused by rules are considered as events that trigger the rules associated with the updated attributes (of the corresponding concepts). Rules thus trigger each other, and this process continues until there are no more triggered rules.
5. A concept may have either just one associated page, or it may have several, one of which is (adaptively) selected, as we explain in section 5 (on adaptive link destinations). The selected page is retrieved (either from local storage or from a remote website) and filtered according to AM and UM, as follows:
  - a. The page may contain conditionally included fragments and/or conditionally included objects. Each selected fragment or object is inserted into the parse stream and must thus be a valid piece of (x)html. It may contain other fragments or objects to be conditionally included.
  - b. The page may contain “conditional” links (<a> tags) to other concepts or pages. The AHA! engine checks UM to decide how to present the link. Link adaptation typically depends on the suitability of the link destination (concept or page). This suitability determines how the link anchor is annotated (using icons and/or link color changes). The section on link adaptation explains the details.

- c. Any other content of the page is passed to the browser. Some text fragments may be adaptively presented using a different presentation style, but most content is normally presented without any adaptation.

It is important to note that this procedure implies that UM updates happen before the page is filtered using that UM instance. Using typical adaptation rules (delivered with AHA! by default), the “knowledge value” of a page is updated *before* the page is displayed. This “knowledge value” can thus not be used to conditionally show a *prerequisite explanation* upon the first visit of the page (and not in subsequent visits). The “visited” counter can be used for this purpose. This counter is initialized to 0 so it will have the value 1 when the page is visited for the first time because it is incremented before the page is presented.

There is a good reason why the UM updates happen *before* the page is filtered and presented, not just in AHA! but in all Web-based adaptive systems that use the HTTP request/response paradigm on a page by page basis. This can be illustrated using prerequisite relationships in an educational application. In elementary math you probably have to learn about the “addition” before you learn about the “subtraction”. “Addition” is thus a *prerequisite* for “subtraction”. Links to “subtraction” will not be shown or recommended until enough knowledge of “addition” is obtained. Now consider that there is a page that provides a lot of information about the “addition” and that contains a link to “subtraction”. The author’s idea is that the user should read this page and then follow the link to “subtraction”. When reading the “addition” page the user will obtain enough knowledge of “addition” to satisfy the prerequisite requirement for “subtraction”. So *after* reading this page the link to “subtraction” should become recommended. But the adaptation (including the presentation of the link to “subtraction”) is done *before* the reading begins. The UM update on which the adaptation is based must thus be performed before presenting the page.

In the future it may become possible to adapt the page (in a controlled way) while the user is reading, so links may become available and recommended during the reading process. Usability research is needed in order to decide whether adaptation while reading would be a desirable adaptation behavior. For now we only consider the typical architecture in which adaptation must be done *before* the engine sends the page to the browser.

### 3 Combining Adaptation with Stable Presentations

The hypermedia course (which prompted the start of the AHA! development) contains a page about URLs and their syntax. It briefly refers to the addressing scheme in Xanadu. Learners who visit the URL page before the Xanadu page see the following piece of text:

In [Xanadu](#) (a fully distributed hypertext system, developed by Ted Nelson at Brown University, from 1965 on) there was only one protocol, ...

Learners who have seen the Xanadu page see a different explanation:

In [Xanadu](#) there was only one protocol, ...

This is a clear and simple form of content adaptation, which Brusilovsky calls the *conditional inclusion of fragments*. There is no discussion that providing such *prerequisite explanations* is a good thing. However, what isn't so clear is whether the learner, who has first seen this fragment, then studies the page on Xanadu and then comes back to the URL page, should be presented the prerequisite explanation *again*, or not. Typical adaptive behavior would remove the explanation on this repeat visit. In this specific example we have not (yet) experienced that learners noticed the ongoing adaptation. But when the changes are more significant it may become desirable to leave a prerequisite explanation (or any content that is conditionally included for some reason) in place even though the adaptation rules dictate that the content is no longer needed.

The on-line adaptive AHA! design paper [11] at <http://aha.win.tue.nl/ahadesign/> makes use of the small elementary math example (about addition and subtraction) we described in section 2. The example appears in two places in that paper. It is *explained* on the first of these pages the reader visits, and is *recalled* on the second page:

In elementary math you probably have to learn about the “addition” before you learn about the “subtraction”. “Addition” is said to be a *prerequisite* for “subtraction”.

or

Recall the elementary math example about “addition” and “subtraction”.

Whichever page is visited first will conditionally include the explanation. If we call the pages “page1” and “page2” then the rule for including the explanation would be:

page1.visited + page2.visited == 1

Afterwards (when the example is visited for a second time, or more) the reader no longer needs the explanation, so the example is simply recalled. But it is not desirable to have the explanation disappear completely on all pages, which is what would happen if the above expression would be used for the adaptation all the time. In order to show the explanation again upon a second visit to *that first visited page* (but not show it on the other page) the presentation of the pages must be made *stable*, meaning that adaptation is performed on the first visit, and no more adaptation is performed when the page is presented again (although the “visited” counters are still incremented).

In AHA! *pages* and *objects* can be defined as stable. When a page is stable that stability is automatically inherited by *included objects*. (At the same time small fragments that are conditionally included *inline* are still adapted, as are the link annotations. The Xanadu example in the hypermedia course is realized using *inline* fragments and thus remains “unstable”.) In AHA! stability comes in four different gradations:

- *no stability*: this is the default behavior where each UM update immediately influences the adaptation, at all times.
- *always stable*: the first time the concept is accessed UM updates are applied, and the concept is presented based on that UM state; afterwards the presentation is still based on that UM state (even though UM updates are still performed and recorded because other, unstable, pages may depend on these updates).

- *session stability*: the first time a concept is accessed during a session, UM updates are applied, and the concept is presented based on that UM state; afterwards the presentation is based on that UM state for as long as the session lasts. When the user logs out and later resumes the session, adaptation is again applied on the first access.
- *conditional stability*: the stability holds as long as a certain condition (an expression using UM attribute values) is true. When the expression becomes false, adaptation is done based on the new UM state. (The page remains unstable until the expression becomes true again.)

Creating stable presentations is very easy. Figure 3 shows the dialog box used by the Graph Author tool to create a new concept or edit an existing one. Stability is enabled through a simple checkbox. When *conditional stability* is selected the author must provide an expression which “freezes” the page as long as that expression remains true.







Fig. 3. Edit concept dialog box that allows the selection of stability

## 4 Adaptive Link Annotation

The (early) Web has been criticized for not having *typed links*. Hyper-G [1], later renamed to Hyperwave, proposed a completely new architecture with links as first-class citizens, but it has not become widely adopted. Using *classes* and *cascading style sheets* (CSS) it is possible to assign a “type” (or “class”) to a link anchor, and to associate presentation attributes with that type. This is of course only a small step towards making links become full-fledged objects, but it is a start. AHA! uses link classes to choose a color for the link anchor (text). An arbitrary number of link classes

can be defined, but in this paper we refer to the three link classes that have been used most in AHA!-based courses up to now: GOOD (recommended), NEUTRAL (recommended but already visited) and BAD (not recommended). There are two ways to associate these to classes to colors:

1. The author can include a (reference to a) stylesheet in every page, defining the presentation style for each link class. AHA! provides a default stylesheet (aha.css) which defines the standard color scheme with blue, purple and black, for good, neutral and bad links. If the author does not include a stylesheet the AHA! engine will insert one automatically.
2. AHA! offers a special form through which end-users can change the color scheme. (This is only available if the author includes access to that form in the course.) When an end-user changes his color preferences AHA! will automatically insert stylesheet commands in the page, overriding any predefined stylesheet.

A number of existing adaptive hypermedia/learning systems also use icons, placed in front of or behind link anchors, to indicate the status of the link (destination). The systems that are descendents of ELM-ART [5] (including Interbook [4] and NetCoach [17]) and other systems simply inspired by it, like KBS-Hyperbook, make use of such icons. A green , white , or red  ball (or yellow or orange in some systems), placed in front of the link anchor, is used to indicate the state of the link and corresponds to the idea of good, neutral and bad links. Sometimes icons behind the link anchors are used, like small , medium  or large  checkmarks to indicate how much knowledge the learner already has about the destination concept. In AHA! a file "ConceptType-Config.xml" contains pairs of expressions and icons, indicating under which condition which icon should be displayed. An example:

```
<icon expr="suitability && visited==0"
      place="front">icons/GreenBall.gif</icon>
```

This tag expresses that when the destination of a link is "suitable" and has not been visited before ("visited==0") then a green ball will be placed in front of the link anchor. Because of the binding of icons to user model expressions the icon selection of different other systems can be easily simulated. When the knowledge levels of Interbook are translated to values like 0, 30, 60, 90 (so that knowledge increments by 1/10 of a level are also possible, as is done in Interbook when reading non-recommended pages), the checkmarks that follow links to (background or outcome) concepts can be chosen using rules like:

```
<icon expr="knowledge > 29 && knowledge < 60"
      place="back">icons/SmallCheckM.gif</icon>
```

AHA! supports *forward* as well as *backward reasoning*, meaning that the attributes like "suitability" and "knowledge" may contain either stored values, calculated through rules that are executed when a concept is accessed, or expressions over (other) attributes of (other) concepts, and thus calculated (backwards) from many different user model values.



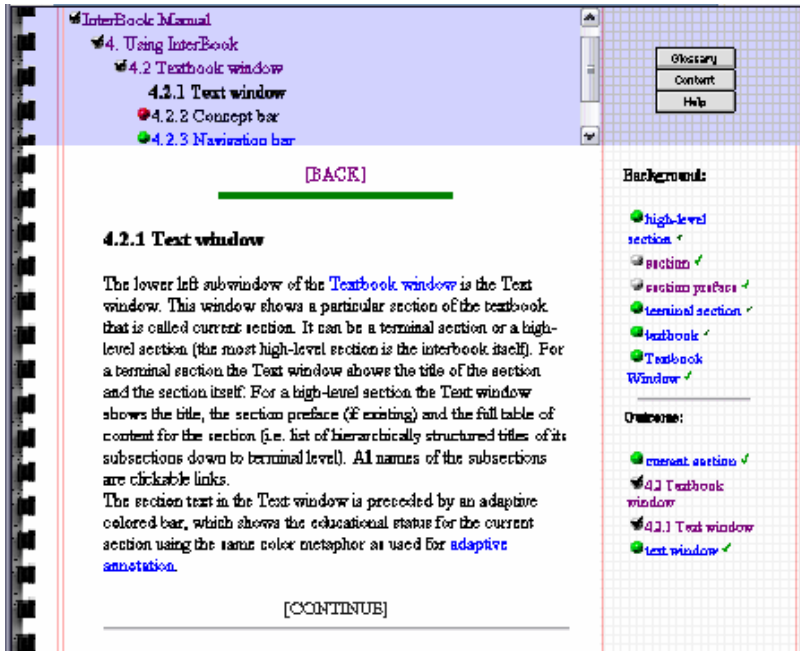


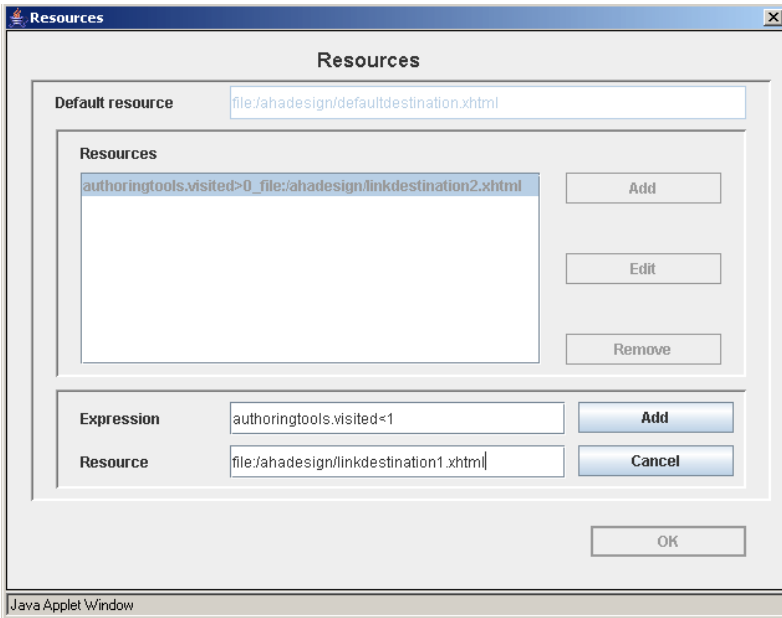
Fig. 4. An AHA! application with adaptive link icon annotations

Figure 4 below shows an AHA! application, generated automatically from the Interbook Manual (which is of course authored for Interbook). This AHA! application shows a clear resemblance to the original Interbook application (including the typical background and the colored balls and checkmarks) but it also uses additional adaptive elements offered by AHA!, including the use of different link colors, not just for links to concepts but also for the “back” and “continue” buttons.

## 5 Adaptive Link Destinations

In a “normal” Website pages contain link anchors, with link destinations identified using uniform resource locators, or URLs. The connections between pages are thus completely fixed. At the other end of the spectrum we find *open hypermedia* (a rich research field with many publications, among others leading to the FOHM model published in [13]), where the pages do not contain any links, but instead are combined with a link database in order to (possibly adaptively) select the links that are going to be shown to the user. AHA! takes an intermediate approach: the author of pages must include the link anchors, but the link destinations are *concepts*, not *pages* (although that is still possible), and the adaptation engine decides which page to show when a link to a certain concept is followed. So this decision is not made when a page (containing links) is generated, but only when a link on the page is followed.

For most concepts and pages there is a one-to-one mapping between a concept and the corresponding page, and vice versa. However, AHA! allows a concept to be associated with multiple pages, one of which is (adaptively) selected for presentation to



**Fig. 5.** Resource selection dialog box

the user. Figure 4 shows the dialog box used by the Graph Author tool for creating a list of expressions coupled with resources (pages). The adaptation engine will return the first resource with an expression that evaluates to true.

This dialog box is the same for *conditional link destinations* and for the *conditional inclusion of objects*, and there is also a potential very similar use of these techniques. Research into learning styles [6] has revealed that *field-independent* learners can start a course by diving right into the details of a single topic without first getting acquainted with the whole spectrum of subjects of the course. They can thus follow a *depth-first* navigation path. *Field-dependent* learners on the other hand need more context, and can be helped by offering them a *breadth-first* navigation path. From this we can conclude that at least for field-dependent learners it is a good idea to provide an introductory page on each major topic, before diving into the details. In a course like the hypermedia course that prompted the development of AHA! there are three introductory chapters that can be studied in any order, and there are six “advanced” chapters that should be studied after the first three. A field-dependent learner would like to at least get a glimpse of these advanced chapters before studying any chapter in detail. There are two ways to make this possible:

1. On a page you can conditionally include objects. An `<object>` tag that refers to a concept is linked to resources in the same way as shown in Figure 5. The resource must be a valid xhtml fragment. (If you wish to conditionally include some media item like an image you have to include it from within the xhtml fragment, e.g. by using an `<img>` tag.) In order to show an introduction to the field-dependent learner who visits the chapter at the start of the learning process and to show the “normal” page (possibly containing links to

pages with more details) to field-independent learners, or to the field-dependent learner who is ready for it, a skeleton page can be created that includes an object that is conditionally connected to the two resources. However, using conditionally included (page-size) objects to simulate having two different pages is a rather artificial use of this technique.

2. Offering an introductory page and a “normal” page can be done by conditionally assigning two pages (or “resources”) to the concept that represents the chapter. Through the dialog box of Figure 5 the conditions for presenting each version are defined. Because the selection is now done entirely in the conceptual definition of the application generating adaptive link destination structures is easier than generating skeleton pages and conditional object inclusions. This is especially important for authoring tools that generate AHA! applications from some high-level description format, possibly including learning style adaptation [14]. Tools that perform such translation are the MOT to AHA! convertor [7, 15] and a learning-style authoring tool of [16] (which has since been extended significantly).

## 6 Discussion and Conclusion

There is no *one size fits all* solution in adaptive (personalized) e-learning. This not only holds for the learners (who have different learning styles) but also for authors. AHA! therefore offers a rich set of possibilities for creating adaptive applications, with different layout, different menu-like structures, different ways to adapt the presentation of links, with or without icons, different ways to adapt the page content, with inline fragments or with external conditionally included objects, and with adaptive link destinations to enable the same link to lead to different pages, all of this depending on the values in the user model.

However, *more* is not always *better* in the area of adaptive hypermedia and personalized e-learning. Too much adaptation may turn an application into an adventure game. In this paper we proposed to use *stable presentations* as an alternative for simply applying less adaptation. Applying adaptation once (and perhaps later again, under controlled circumstances) makes the adaptation invisible to the users because they only see one version of each page. They may only become aware that the application is adaptive when they compare what they see on a page to what other users see on the same page.

Having a rich functionality for creating adaptive behavior, and ways to disable unwanted adaptation, does not guarantee that usable adaptive applications will be produced. The Graph Author tool lets authors create a conceptual structure in a graphical way. An author can for instance simply draw a graph of prerequisite relationships between concepts. And concepts automatically become part of a hierarchy of larger concepts (like chapters) and smaller ones (like pages and fragments). But this conceptual structure must still be *designed* carefully. A drawing tool (with additional features such as cycle detection) for concept relationships of different types is a useful tool but it does not perform the design phase for you.

People have frequently asked us whether creating an adaptive course using AHA! is more work (and if so, how much more) than creating a static Web-based course.

The answer is twofold: yes, it is more work, because one has to consider *prerequisites* and design measures to deal with them. This includes creating the prerequisite relationships in the Graph Author tool so that links are properly annotated (to guide users), and it includes writing *prerequisite explanations* to conditionally include on pages that require prerequisite knowledge that can be sufficiently compensated for by a short explanation (instead of warning the learner not the page at all), or writing introductory pages to replace detailed ones, and have links conditionally refer to the intro or the details. On the other hand, creating a non-adaptive (or static) Web-based course text that can be browsed freely without encountering pages that cannot be understood because of missing foreknowledge is completely impossible. So we argue that it is a matter of putting in some extra effort to create a course of high quality that cannot be obtained without using adaptation. AHA! tries to minimize the required extra effort but cannot completely eliminate it.

## Acknowledgements

This work is/was supported by the PROLEARN network of excellence and the NLnet Foundation.

## References

1. Andrews, K., Kappe, F., Maurer, M., Serving Information to the Web with Hyper-G. Third International World Wide Web Conference, Computer Networks and ISDN Systems (27) pp. 919-026 (1995).
2. Brusilovsky, P., Methods and Techniques of Adaptive Hypermedia. User Modeling and User-Adapted Interaction, 6, pp. 87-129, 1996.
3. Brusilovsky, P., Adaptive Hypermedia. User Modeling and User-Adapted Interaction, 11, pp. 87-110, 2001.
4. Brusilovsky, P., Eklund, J., Schwarz, E., Web-based education for all: A tool for developing adaptive courseware. Computer Networks and ISDN Systems (Proceedings of the 7th Int. World Wide Web Conference, 30 (1-7), pp. 291-300, (1998).
5. Brusilovsky, P., Schwarz, E., Weber, G., ELM-ART: An intelligent tutorial system on World Wide Web. In Proceedings of ITS'96, Intelligent Tutoring Systems (Springer LNCS Vol 1086), pp. 261-269 (1996).
6. Chen, S., Macredie, R., Cognitive styles and hypermedia navigation: Development of a learning model. Journal of the American Society for Information Science and Technology, 53 (1), pp. 3-15 (2002).
7. Cristea, A.I., Smits, D., De Bra, P., Writing MOT, Reading AHA! - converting between an authoring and a delivery system for adaptive educational hypermedia. A3EH Workshop, AIED'05 (2005).
8. De Bra, P., Hypermedia Structures and Systems. Adaptive course text offered at the TU/e, available at <http://www.wis.win.tue.nl/2L690/> (1994, 1996).
9. De Bra, P., Aerts, A., Berden, B., De Lange, B., Rousseau, B., Santic, T., Smits, D., Stash, N., AHA! The Adaptive Hypermedia Architecture. Proceedings of the ACM Hypertext Conference, Nottingham, UK, pp. 81-84 (2003).
10. De Bra, P., Aerts, A., Rousseau, B., Concept Relationship Types for AHA! 2.0. Proceedings of the AACE ELearn'2002 conference, pp. 1386-1389 (2002).

11. De Bra, P., Smits, D., Stash, N., The Design of AHA!. Proceedings of the ACM Hypertext Conference, Odense, Denmark (2006), and on-line adaptive version at <http://aha.win.tue.nl/ahadesign/>.
12. Henze, N., Nejd, W., Adaptivity in the KBS Hyperbook System. Second Workshop on Adaptive Systems and User Modeling on the WWW, TU/e CSN 99-97, pp. 67-74, Toronto, Canada, (1999).
13. Millard, D., Moreau, L., Davis, H., Reich, S., FOHM: A Fundamental Open Hypertext Model for Investigating Interoperability between Hypertext Domains. Proceedings of the ACM Conference on Hypertext, pp. 93-102 (2000).
14. Stash, N., Cristea, A., De Bra, P. Explicit Intelligence in Adaptive Hypermedia: Generic Adaptation Languages for Learning Preferences and Styles, HT'05, CIAH Workshop, Salzburg, (2005).
15. Stash, N., Cristea, A.I., De Bra, P., Authoring of Learning Styles in Adaptive Hypermedia: Problems and Solutions, WWW'04 (The 13th International World Wide Web Conference) pp. 114-123 (2004).
16. Stash, N., De Bra, P., Incorporating Cognitive Styles in AHA! (The Adaptive Hypermedia Architecture), Proceedings of the IASTED International Conference Web-Based Education, pp. 378-383 (2004).
17. Weber, G., Kuhl, H.-C., Weibelzahl, S., Developing Adaptive Internet Based Courses with the Authoring System NetCoach, Proceedings of the Third Workshop on Adaptive Hypermedia (AH2001), Springer LNCS Vol. 2266, pp. 226-238 (2001).