AHA! The Adaptive Hypermedia Architecture

Paul De Bra, Ad Aerts, Bart Berden, Barend de Lange, Brendan Rousseau,
Tomi Santic, David Smits, Natalia Stash
Department of Computer Science
Eindhoven University of Technology
Eindhoven, The Netherlands
+31 40 2472733
debra@win.tue.nl

ABSTRACT
AHA!, the "Adaptive Hypermedia Architecture", was originally developed to support an on-line course with some user guidance through conditional (extra) explanations and conditional link hiding. This paper describes the many extensions and tools that have turned AHA! into a versatile adaptive hypermedia platform. It also shows how AHA! can be used to add different adaptive "features" to applications such as on-line courses, museum sites, encyclopedia, etc. The architecture of AHA! is heavily inspired by the AHAM reference model [6].

ACM Categories
H5.2 (user interfaces), H5.4 (hypertext/hypermedia)

General Terms: Design, Experimentation, Human Factors

Keywords: Adaptive hypermedia, adaptive presentation, adaptive navigation support, authoring support.

1. INTRODUCTION
After some initial experimental versions AHA! [7] was released as version 1.0 in 2000 [5]. Compared to other adaptive systems like Interbook [2], KBS-Hyperbook [8,10] and many others AHA! excelled in the area of simplicity. From the many adaptive hypermedia methods and techniques presented in Brusilovsky's taxonomy [1] AHA! version 1.0 offered only the most basic low-level elements:

- *A simple user model based on pages and concepts*: Each time you visit a page a user model value (integer between 0 and 100) is updated. The update can (in some cases) be "propagated" to other user model entities (pages or abstract concepts). It is easy to define the user model updates for representing knowledge in a concept hierarchy (like for studying a textbook) where pages contribute knowledge to topcs/sections, sections to chapters and chapters to the "root concept" or whole textbook. However, AHA! lets you use the user model updates for any other purpose as well.
- *Adaptive link hiding or link annotation*: The *suitability* of pages is determined by an author-defined requirement. This is a Boolean expression using arbitrary user model values. The requirements can express the common *prerequisite relationships* between concepts but can be used for any other condition that can be expressed through such a Boolean expression. When a page is generated, links marked as *conditional* (using the link class "conditional") are displayed differently depending on the evaluation of the expression for the link (destination). If the expression is true the link is shown in blue (unvisited) or purple (visited), and when the expression is false the link is shown in black, and not underlined. This results in hiding the *unsuitable* or *undesired* links. The color scheme can also be altered so that links are all visible, in different colors.
- *Conditional inclusion of fragments*: Like for the links to pages the author can also associate a requirement with fragments in a page. This is done through an <if> tag, with one or two fragments, enclosed by a <block> tag. If the expression is true the first fragment is shown to the user, otherwise the second (optional) fragment is shown. This can be used to include *prerequisite explanations*, or any other piece of content.

AHA! version 1.0 was not only very simple, it was also delivered as Open Source software, written entirely in Java, using Servlets. Several researchers in different countries have used AHA! either for research projects or for courses (or both) [4,9,11]. While the basic functionality of AHA! has been retained, many extensions have been proposed and realized. In this paper we present new features, inspired by the AHAM reference model [6], by functionality of other systems and by experience with (and shortcomings of) AHA! version 1.0. We present AHA! 2.0 and 3.0 functionality, not as a list of additions, but through a description of AHA! as a whole, thereby enabling the reader to get a good idea of the complete functionality of AHA! as well as of how AHA! can be used to create adaptive applications.

In Section 2 we describe the overall architecture of AHA!. This is followed in Section 3 by a description of how the user's interaction with the system results in user model updates, through the *adaptation model*. Section 4 describes the adaptive presentation and the link adaptation functionality. Section 5 deals with the possibilities to keep parts of a presentation stable. Section 6 introduces the authoring tools. In the final Section 7 we provide an outlook into the future.

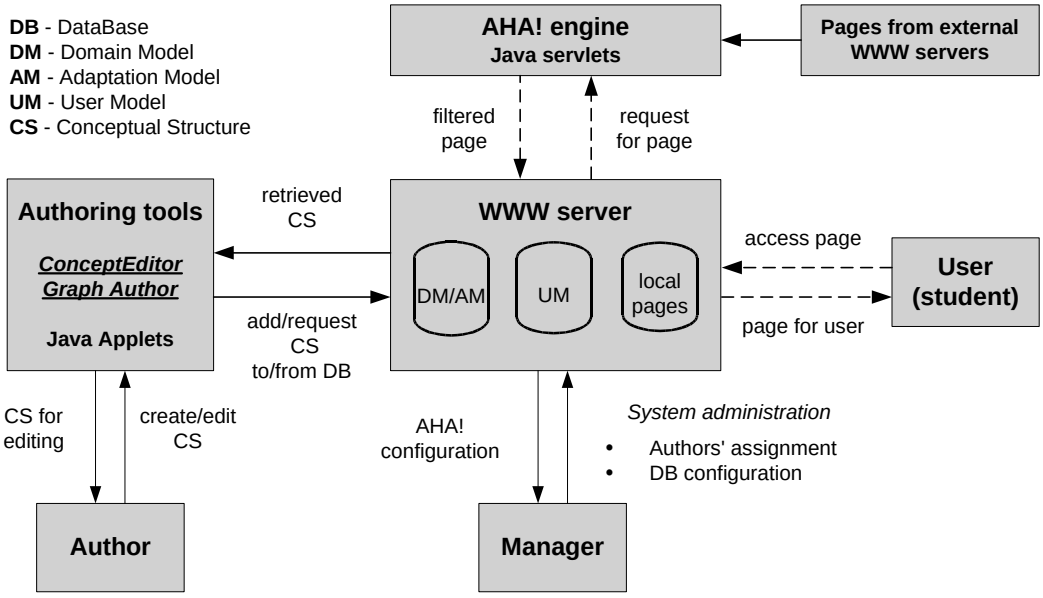

Figure 1: Sketch of the architecture of AHA!

2. ARCHITECTURE OF AHA!

AHA! (all versions) consists of Java servlets that mainly serve pages from the local file system or from external http servers. The servlets interact with a combined *domain/adaptation model* DM/AM and with a *user model* UM (in terms of AHAM [6]). A request for a page (passed to a servlet by the webserver) triggers adaptation rules that perform UM updates. When UM is updated the requested page is parsed to perform the conditional inclusion of fragments. That inclusion is based on the new state of UM. Links on the page (or in included fragments) may be "conditional", meaning that their presentation depends on a "suitability" requirement that is part of the domain/adaptation model.

AHA! stores DM/AM and UM (of all users) either as XML files or in a mySQL database. The choice between these two is made by the Manager who configures AHA!, chooses installation directory, path names, etc., and who creates accounts for authors. The configuration is done entirely through a web-based forms interface. AHA! comes with installation instructions for its use with the Open Source Tomcat server.

Authors typically create the domain/adaptation model through an authoring tool. The Concept Editor offers low-level access to the adaptation model's rules, whereas the Graph Author is used to define concept relationships such as *prerequisites*. Both tools are described in Section 6. Alternative authoring tools are being developed. Recently, a compiler from Interbook [3] to AHA! was developed, along with an AHA! addition called the *layout model*, to create presentations that consist of multiple windows and frames. Another experimental authoring tool uses a scripting language to describe pages and the possible navigation paths [9]. It was defined as part of an attempt to combine (parts of) AHA! with Auld Linky, an Open Hypermedia system developed at the University of Southampton.

Figure 1 sketches the overall architecture of AHA!.

3. DM/AM AND UM IN AHA!

AHA! applications mainly consist of a set of *concepts*, some of which are linked to *pages* or *objects* (or *fragments*). Concepts can be used to represent *topics* of the application domain, e.g. subjects to be studied in a course, or artists, art styles, or art pieces (like paintings) in a museum. In AHA! the author of an application can associate any number of (named) attributes with a concept. Some attributes have a meaning for the system, like *access* (a Boolean attribute that temporarily becomes true when a page is accessed), some have meaning for the author and user, like *knowledge* or *interest*, and some have meaning for both, like *visited* (determining the blue versus purple link color). Since AHA! uses an *overlay user model*, all attributes of concepts in DM/AM also appear in UM.

The adaptation rules define how the user model is updated. When the user accesses a page (or an object included in a page) the rules associated with the *access* attribute are triggered. Each rule consists of two parts: a *condition* and an *action*. The condition is expressed as a Boolean expression using attributes of concepts, and the action consists of one or more assignments of values or expressions to attributes of concepts. (The expressions can use constants, attribute values, or even the amount by which the triggering attribute was changed.) Optionally there can also be a second action, that is performed when the condition is false.

The actions of a triggered rule update some attributes of some concepts. The author can indicate for every rule whether this action triggers the rules associated with the updated concept-attributes. It is thus easy to define that a page access increases the *knowledge* of the concept associated with the page, and have that knowledge update cause a knowledge update of a larger topic, like a section of a textbook, and have that knowledge update in turn cause a knowledge update of an even larger topic like a chapter. Similarly, accessing a page that shows a painting in a museum can cause an *interest* update for the painter, and that may cause an interest update for the style of that painter, etc. The fact that these adaptation rules are completely arbitrary illustrates the power of

the AHA! system, but at the same time also shows that it is difficult for an author to predict the outcome of the rule execution. It is even possible that the defined rules sometimes cause an infinite loop in the rule execution. (AHA! does limit the rule execution to avoid run-time errors in such a case.) In Section 6 we briefly introduce the *Graph Author* tool that enables authoring without the need to be familiar with the low level adaptation rules.

An aspect worth mentioning is the way in which AHA! determines the *suitability* of a page. With each page (or concept) AHA! associates a *requirement expression*. This expression is evaluated when needed to determine the suitability of a page. However, rather than evaluating the suitability when needed one can also opt to use a *suitability attribute*, and just evaluate that attribute in the expression. One can thus determine the suitability of a page through adaptation rules.

Another aspect is the *visited* status of pages or objects. AHA! uses a *visited attribute* to store this status. When a page or object is accessed an adaptation rule can set the *visited* attribute. This is done using a "standard" rule, normally created automatically by the authoring tools. However, since it is just a user model attribute of the concept one can create rules that manipulate this attribute in a different way (e.g. marking previously visited pages as unvisited for some purpose).

4. HOW AHA! PERFORMS ADAPTATION

Adaptation in AHA! is based on a number of attributes associated with concepts, and stored in UM. We distinguish the adaptation of links and of fragments.

- When a page (or fragment) contains a link anchor, and the link is marked as (being of class) *conditional*, the link can be presented using one of three colors, called *good*, *neutral* and *bad*. First the *requirement expression* of the link destination (page) is evaluated. If the expression evaluates to false the link anchor is shown in the *bad* color, which is black by default. If the expression evaluates to true AHA! then looks at the *visited* attribute of the link destination. If visited is 0 the link is shown in the *good* color, blue by default, otherwise it is shown in the *neutral* color, purple by default. Whereas this behavior was already present in the old AHA! version 1.0 the author now has the ability to manipulate the way AHA! deals with the link adaptation because it is based on an arbitrary expression (or *suitability* attribute) and a *visited* attribute that can also be assigned values in an arbitrary way.
- Content adaptation in AHA! uses the *conditional inclusion of fragments* technique. There are two ways to use this technique in AHA!: with *embedded fragments* or with *objects*. Embedded fragments appear within a page, and are included if an associated *suitability expression* evaluates to true. This feature is essentially the same as in AHA! 1.0, but the expressions can be more complex because of the availability of arbitrarily many attributes for each concept. The main disadvantage of this technique is that it mixes a DM/AM construct with the actual content pages. Therefore it is better to use the AHA! 3.0 feature of conditional inclusion of *objects*. In a page the author includes <object> tags, with objects of a special type "aha/text". The AHA! engine looks at the ID of the object, which is considered as a concept from DM. The *access* event of that concept is used to trigger the adaptation engine and perform UM updates. An attribute can be associated with a *casegroup* tag to select the resource (file) to be inserted in the page at the position of the <object> tag. That fragment is called the *returnfragment*. It is thus not only possible to conditionally include "a" fragment, but also to choose *which* fragment to include. Parsing of the page continues with the contents of the fragment (if it is an XHTML file like the page). The fragment may also contain an <object> tag, which again triggers the adaptation engine. This process (which might run into an infinite loop if the inclusion is recursive and badly conditioned) continues until all fragments are considered for inclusion and the end of the page is reached.

When the adaptation is done based on conditional links and on objects, the pages can be standard XHTML, which means that standard authoring tools can be used to create them. Also, other XML formats can be used as long as they have an <a> tag and an equivalent of the <object> tag. SMIL has been used successfully, using the <ref> tag instead of <object>. AHA! can thus be used to augment the limited adaptive functionality already foreseen in SMIL.

5. STABLE PRESENTATIONS

In a typical adaptive application each page is adapted to the current UM state each time that page is visited. This may not be the desired behavior according to some users. It may be unsettling that an adaptive website does not always show the same information on the same page.

For each page or object in AHA! 3.0 one can indicate the desired *stability* of the presentation. Each page or object can be adapted in four different ways:

- *always adapted*: this is the default behavior, in which the page or object is always adapted to the current UM state.
- *always stable*: on the first access the page or object is adapted to the UM state. On subsequent accesses the presentation is identical to the first presentation (except for the link colors based on the visited state, like users have come to expect in web browsers).
- *session stable*: on the first access during a session the page or object is adapted to the UM state. During the session the presentation remains the same.
- *expression stable*: a <stable_expr> is associated with the page or object. As long as this expression remains true the presentation is kept stable. When the expression becomes false and the page or object is accessed again it is again adapted to the current UM state.

In AHA! it is possible to include the same object multiple times on a page (each time possibly resulting in a different fragment). Defining stability for objects only would not allow such pages to be presented in a stable way. Object stability and page stability are therefore both included in AHA!. However, we expect that situations with multiple instances of the same object on a page will be rare.

6. AUTHORING TOOLS

AHA! version 1.0 was successful mainly thanks to its simplicity. Writing AHA! applications was relatively easy. It did require writing some XML files by hand, but the format was simple and not very verbose.

Since AHA! version 2.0 (and thus also with the upcoming 3.0) the DM/AM structure has become a lot more complicated: for each

concept there are arbitrarily many attributes. Each attribute can have condition-action rules to update arbitrary attributes of arbitrary concepts. (In AHA! version 3.0 there are even more features, like the *casegroup* to choose fragments to include, *stability* parameters, etc.) Creating AHA! applications without an authoring tool to create the DM/AM is no longer feasible. Authoring for AHA! is facilitated in three ways:

- The *Concept Editor* is a graphical, Java applet based tool to define concepts and adaptation rules. It uses an (author-defined) template to associate a predefined set of attributes and adaptation rules with each newly created concept. It is a *low level* tool in the sense that all adaptation rules between concepts must be defined by the author. Many applications have a number of constructs that appear frequently, e.g. the knowledge propagation from page to section to chapter, or the existence of prerequisite relationships. This leads to a lot of repetitive work for the author.
- The *Graph Author* is also a graphical, Java applet based tool, but it uses *high level* concept relationships. Again, when concepts are created a set of attributes and adaptation rules is generated. But this tool also has templates for different types of concept relationships (also defined by the author). Creating knowledge propagation, prerequisite relationships or any other relationship is just a matter of drawing a graph structure using this graphical tool. The translation from high-level constructs to the low-level adaptation rules is done automatically.
- Instead of creating AHA!-specific tools it is possible to let authors develop applications for other adaptive hypermedia systems and translate them to AHA!. Given the fact that AHA! offers a lot of (low-level) functionality this should be possible for many systems. As part of the work on additions to the presentation and layout possibilities of AHA! an Interbook to AHA! compiler was developed [3]. It shows that AHA! can not only emulate the adaptive behavior of Interbook, it can even mimic the presentation quite closely.

7. CONCLUSIONS AND FUTURE WORK

AHA! has been extended in many ways in the past two years. This paper has described the main features of the latest developments (AHA! versions 2.0 and 3.0). The result is a web-based system that:

- uses standard XHTML for pages and for included fragments, thus enabling the use of off the shelf authoring tools;
- has a rich user model, representing abstract concepts, pages and fragments, with arbitrarily many attributes per concept, as suggested by the AHAM reference model [6];
- offers a simple form of link adaptation, closely resembling the behavior users expect in a browser environment;
- offers a powerful content-adaptation method, with in version 3.0 the ability to conditionally select objects to include, and enabling the use of the same object on different pages without redundant copies.

Feedback from real users will be the most important source of inspiration for new developments. However, two important ideas have already emerged:

- There is a need to enable adaptive hypermedia systems to communicate with each other. This means access to each others' DM/AM as well as UM, through a common protocol.
- Adaptation has until now been based on the user's browsing behavior. Work is needed to also allow adaptation to aspects of the user, such as cognitive style, and the user's context, such as device and network characteristics.

8. ACKNOWLEDGEMENT

The development of the AHA! system is supported by a grant of the NLnet Foundation, through the "Adaptive Hypermedia for All!" project (conveniently abbreviated to AHA!).

9. REFERENCES

[1] Brusilovsky, P. Adaptive Hypermedia, User Modeling and User-Adapted Interaction, Vol. 11, nr. 1–2, pp. 87–110, Kluwer academic publishers, 2001.

[2] Brusilovsky, P., Eklund, J., and Schwarz, E. Web-based education for all: A tool for developing adaptive courseware. Proceedings of Seventh International World Wide Web Conference, pp. 291-300, 14-18 April 1998.

[3] Brusilovsky, P., Santic, T., De Bra, P. A Flexible Layout Model for a Web-Based Adaptive Hypermedia Architecture. Proceedings of the AH2003 Workshop, TU/e CSN 03/04, pp. 77-86, Budapest, Hungary, May 2003.

[4] Cini, A., Valdeni de Lima, J. 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, May 2002.

[5] De Bra, P., A. Aerts, G.J. Houben, and H. Wu. Making General-Purpose Adaptive Hypermedia Work. Proceedings of the AACE WebNet Conference, pp. 117–123, San Antonio, Texas, 2000.

[6] De Bra, P., G.J. Houben, and H. Wu. AHAM: A Dexter-based Reference Model for Adaptive Hypermedia. Proceedings of the ACM Conference on Hypertext and Hypermedia, pp. 147–156, Darmstadt, Germany, 1999.

[7] De Bra, P. and Calvi, L., AHA! An open Adaptive Hypermedia Architecture. The New Review of Hypermedia and Multimedia, vol. 4, pp. 115-139, Taylor Graham Publishers, 1998.

[8] Henze, N. Open Adaptive Hypermedia: An approach to adaptive information presentation on the Web. First International Conference on Universal Access in Human-Computer Interaction (UAHCI 2001), held jointly with HCI International 2001, 5-10 August 2001, New Orleans, Louisiana, USA.

[9] Millard, D., Davis, H., Weal, M., Aben, K., De Bra, P. AHA! meets Auld Linky: Integrating Designed and Free-form Hypertext Systems. Proceedings of the ACM Hypertext Conference, Nottingham, UK, August 2003.

[10] Nejdl, W. and Wolpers, M. KBS Hyperbook -- A Data-Driven Information System on the Web. WWW8 Conference, Toronto, May 1999.

[11] Romero, C., Ventura, S., De Bra, P., De Castro, C. Discovering Prediction Rules in AHA! Courses. Proceedings of the User Modeling Conference, Johnstown, Pennsylvania, June 2003.