Authoring and Management Tools for Adaptive Educational Hypermedia Systems: the AHA! case study

Paul de Bra[1], Natalia Stash[1], David Smits[1],
Cristóbal Romero[2], Sebastián Ventura[2]
[1] Eindhoven University of Technology (TU/e),
PO Box 513, Eindhoven, The Netherlands
{debra}@win.tue.nl
[2] Córdoba University, Campus Universitario de Rabanales,
14071, Córdoba, Spain
{cromero, sventura}@uco.es

Abstract. Creating and maintaining adaptive educational applications is a hard work for teachers and developers. In order to help the author to perform these tasks the e-learning systems must provide authoring and management tools. In this chapter we are going to describe several useful tools for working with adaptive educational hypermedia systems, concretely with the AHA! system. AHA! is an well-known open source general-purpose adaptive hypermedia system. In the current AHA! distribution version there are some general adaptive author tools as Concept Editor, Graph Editor and Form Editor, all accessible through the overall Application Management Tool. There is also a specific educational tool: the Test Editor (and the associated Test Engine) and we are now developing some others such as a Course Editor and a Mining tool. In this chapter we are going to describe the AHA! system and the functionality of each of these authoring and management tools intended to help to the teachers and application developers.

1 Introduction

In the past years, we have seen an explosive growth in the use of web-based technology in distance learning systems. At the same time, more and more artificial intelligence techniques have been integrated into these systems to improve students' learning, turning them into Intelligent Tutoring Systems. The union of web-based hypermedia with Intelligent Tutors has led to the current Web-based Adaptive Educational Hypermedia Systems that allow adapting the teaching to each individual student [10]. Adaptive Educational Hypermedia (AEH) is an alternative to the traditional one-size-fits-all approach in web-based education. It combines the concepts of hypertext/hypermedia with user modeling and user adaptive systems to adapt the hypertext to the needs of each particular student. Adaptive hypermedia systems (AHS) [7] started to appear around 1990, when researchers combined the concepts of hypertext/hypermedia with user modeling and user adaptation. The first and foremost ap-

plication of adaptive hypermedia was in education, where the navigational freedom of hypermedia was introduced into the area of intelligent tutoring systems. But since then applications in information systems, information retrieval and filtering, electronic shopping, recommender systems, etc. have been realized. The advent of the Web has made the use of (basic) hypermedia facilities easier, through the use of HTML. However, creating adaptive hypermedia on the Web requires server-side functionality for user modeling and for the adaptive generation of (HTML) pages. Until recently, almost every adaptive hypermedia application was based on a special-purpose (server-side) system. The development of adaptive hypermedia applications and systems has had a one-to-one relationship. This has seriously hindered the development of interesting new adaptive applications by researchers with insufficient skills or financial means to develop their own adaptive hypermedia system. The potential benefits of adaptive educational hypermedia for educational applications should do that adaptive hypermedia were so wide-spread and widely accepted as we would expect. One of the reasons can be due to the difficulties in authoring. So, it is necessary to provide authoring and managements tools in order to help to the teacher or application developer in performing a multitude of authoring tasks.

Nowadays, there are several modern authoring systems that can used by non-programming to develop adaptive educational hypermedia [11]. Authoring tools can be toolkits, systems or shells. There are two major approaches used by authoring tools: the markup language that uses a special authoring user interface. The main task of these tools is to help to the developer to author the content objects and to specify the links between them. However, in order to maintain an AEH application it is necessary to perform many more tasks such as to develop assessments and activities to evaluate students, to visualize and analyze the students' usage information, to modify and improve the course content and structure, to do back-up and to reuse educational material of other systems, etc. So, it is necessary to provide authoring and management tools for each of these tasks.

A well known AHS system is AHA! [19] [20] [22] [23], or *Adaptive Hypermedia Architecture*. It was designed and implemented at the Eindhoven University of Technology, and sponsored by the NLnet Foundation through the Adaptive *Hypermedia for All*. (AHA!) project. AHA! is an open source general-purpose adaptive hypermedia system, through which very different adaptive applications can be created. AHA! offers low-level facilities for creating exactly the desired look-and-feel for each application and for fine-tuning the adaptation, and it offers high-level facilities for creating the conceptual structure of an application, using concepts and concept relationships. Since AHA! is essentially an adaptive client and server at the same time it can be used as a component in the content delivery pipeline and thus integrated into other server environments. AHA! was originally developed to support an on-line course with some user guidance through conditional (extra) explanations and conditional link hiding. But now AHA! has many extensions and tools that have turned it into a versatile adaptive hypermedia platform and it has a complete set of authoring tools (Concept Editor, Graph Author, Form Editor, Test Editor, etc.) to allow authors to easily create or change applications or courses, concepts, concept relationships, computerized tests, etc.

In this chapter, we are going to describe the AHA! system and the main authoring tools that it provides to the author. First, we describe a state of the art in adaptive educational hypermedia systems and authoring tools for developing. Then, we describe overall AHA! architecture and the tools that are included in the current distribution version, available from the Eindhoven University of Technology, at http://aha.win.tue.nl/. Next, we describe some news tools that we are developing at Córdoba University, and finally, we present some conclusions.

2 Background

Adaptive Web-based educational systems (AWBES) provide an alternative to the traditional "just-put-it-on-the-Web" approach in the development of Web-based educational courseware [12]. AWBES attempt to be more adaptive by building a model of the goals, preferences and knowledge of each individual student and by using this model throughout the interaction with the student in order to adapt to the needs of that student. For example, a student will be given a presentation that is adapted especially to his or her knowledge of the subject and a suggestion is made of the most relevant links to proceed further. AWBES inherit from traditional Intelligent Tutoring Systems (ITSs) and Adaptive Hypermedia Systems (AHSs). ITSs typically partition the information space in knowledge about the domain, knowledge about the user and teaching strategies to support individualized learning. Adaptive Hypermedia Systems usually enable content and navigation adaptation, by altering the link structure and the node contents of the hypertext that contains the educational material. AWBES can use different techniques and methods in order to add adaptive functionality to and educational system [7]:

- Curriculum Sequencing (or instructional planning): Provide the learner with the most suitable individually planned sequence of knowledge units and learning tasks.
- Intelligent analysis of a student's solutions: Identify in the student's solution of a problem what exactly is wrong or incomplete and which missing or incorrect knowledge may be responsible for the error.
- Interactive problem solving support: Provide the student with intelligent help on each step of the problem solving process from giving a hint to executing the next step for the student.
- Example-based problem solving: Help students by suggesting the most relevant cases (examples previously explained or problems already solved by the students).
- Adaptive presentation technology: Adapt the content of a hypermedia page to the user's goals, knowledge and other information stored in the user model.
- Adaptive collaboration support: Use the system's knowledge about different users (stored in user models) to form matching collaboration groups.
- Adaptive navigation support technology is to support the student navigation and orientation in hyperspace by changing the appearance of visible links. In particular, the system can adaptively sort, annotate, or partly hide the links in the current page to make easier the choice of the next link to proceed.

There are a lot of adaptive educational hypermedia systems developed since 1996 [10] such as ELM-ART [6], InterBook [8], PT [29], 2L670 [18], Medtech [24], AST [45], ADI [44], HysM [30], MetaLinks [36], RATH [27], TANGOW [14], Arthur [25], CAMELEON [31], KBS-Hyperbook [26], SKILL [37], ACE [46], ART-Web [51], AHA! [19], etc. After some initial experimental versions AHA! was released as version 1.0 in 2000. Compared to other adaptive systems like Interbook, KBS-Hyperbook and many others AHA! excelled in the area of simplicity. AHA! has since evolved into a much more powerful system (version 2.0 and current 3.0), but new versions maintain that basic simplicity. The adaptive hypermedia methods and techniques present in AHA! [20] can be found in Brusilovsky's taxonomy [7]:

- A user model based on concepts: Each time you visit a concept in an AHA! application the name of the concept is passed to the adaptation engine which updates the user model. A user model consists of concepts that have attributes. A typical example of an AHA! action is that visiting a concept may increase a knowledge attribute for that concept. This knowledge update may propagate to the knowledge attribute of other concepts, perhaps corresponding to a section or chapter of a textbook. In AHA! a concept can have arbitrarily many attributes of types Boolean, integer or string. A concept may also have an associated *resource* which is a page to be presented to the user. (A concept may have several associated resources and rules to select the most appropriate page to be presented.)
- Adaptive link hiding or link annotation: The suitability of link destinations (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 suitability of 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 by the end-user to make all links visible, in different colors.
- Conditional inclusion of fragments: Like for the links to concepts or 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. In the current AHA! version 3.0 fragments can be external objects, represented through the <object> tag. Such objects can themselves also be associated with concepts and accessing them triggers user model updates just like for page accesses.

Presently, adaptive educational hypermedia is at the point of break-through from academic circles to industry, attracting more and more business partners. Therefore, the time is ripe to move from the current proof-of-concept type of research to the wide-range, scalable implementations [15]. For this purpose, it is necessary to give

much more attention to the authoring process itself and to produce good authoring tools [16]. Using these tools educators can safely become adaptive educational hypermedia authors, without needing programming skills or excessive training. Generally, the basic steps of creating AWBES are not entirely different from steps in creating a regular hypertext system. In fact, authoring for adaptive educational hypermedia systems means initially creating the resources, labelling them, combining them into what is known as (in adaptive hypermedia) a domain model. Then, another (not necessarily successive) step is creating a user model, responsible for characterizing the user, either in a static way (which generates adaptable material) or in a dynamic way (for adaptive material). More advanced adaptive hypermedia systems dedicated to education also add a pedagogic model, and sometimes a presentation model. As in the case of generic hypertext and Web authoring, there are two major approaches used by authoring tools [11]:

- Markup approach. In this approach the content of the pages and links between pages and concepts are authored in a regular word processor with the help of special markup language. A good example is to save a Word file in RTF format and then to convert it to a real markup representation in the form of extended HTML. Nowadays, XML offers a standard way of markup-based authoring supported by various XML editors and parsers. The markup approach is very attractive because the combination of low cost and expressive power.
- GUI (Graphic user interface) approach. In this approach authors are created applications using a special authoring user interface. In general, it could be a command-based, form-based or a direct manipulation interface, but all existing AHS authoring tools use form-based GUI. This kind of interface provides very good support for the non-professional. Unfortunately, there are no form-based interfaces that can be used to develop rich information structures. So, a combination of a markup-based interface and a form-based interface can provide a good compromise between development cost and level of author's support.

There are several examples of real authoring systems to develop adaptive educational hypermedia [11] such as InterBook [8], Web DCG [48], DCG+GTE [49], ACE [46], ALE [47], NetCoach/ART-WEB [52], ECSAIWeb [42], MetaLinks [36], SIGUE [13], CAMELEON [31], PaKMaS [Süß2000], RATH [27], SKILL [37], ITMS [34], WHURLE [5], AHA! [19], etc. Most of these systems use only one of the two previous authoring approaches. However, AHA! is a good example about a combined interface. While the full power of AHA! (especially in version 3.0) is only available through the XML-based markup language, the system now also provide several form-based authoring tools for the development of domain model (Concept Editor, Graph Editor, Form Editor and Test Editor) and management (Mining Tool and Course Editor). In AHA! each page is created as an XML file that includes specially tagged index part and a page content. The page content is created in HTML format with XML extensions that are used to author conditional fragments (fragment indexing). In addition to that, a special XML file defines domain concepts and connections between them. The domain model XML file lets the author to specify these relationships between concepts. The runtime engine of AHA! based on Java servlets reads the authored XML files, converts it into an internal object representation and

maintains the adaptive interaction with the user. In fact, it is the use of the flexible XML approach that allows AHA! to evolve so rapidly.

Currently, there is a new wave in Web-based courseware authoring centered on learning objects, courseware reuse, metadata and standards [11] that comes from Learning Management Systems (LMSs) in order to resolve problems that one can't move a web-based course from one systems to another, can't reuse web-based content pieces (objects) across different systems, and can't create searchable learning content libraries or media repositories across different environments. One solution is to create eLearning specifications and standards and make them available as free downloads. eLearning standards and specifications refer to a system of common rules for content, authoring software and LMSs. These standards specify how courses can be created and delivered over multiple platforms so that they all operate seamlessly together. Accredited standards ensure that the investment in time and intellectual capital could move from one system to the next. There are several adopted or coming eLearning standards:

- AICC [1] stands for Aviation Industry CBT Committee. It is considered the oldest eLearning standard in the world, originating from the needs of the aviation industry to create a common CBT system. Subsequently, the standard was shifted to encompass web-based training.
- IMS [28] is another popular eLearning standard, focusing mostly on metadata, such as metadata for the tagging of learning objects. It also has specifications to define how the LMS can communicate with back-end applications. The most widely acknowledged IMS specifications are as follows: IMS Meta-data, IMS Content Packaging and IMS QTI (Question and Test Interchange).
- IEEE Learning Technology Standards Committee (LTSC) provides the Learning Object Metadata (LOM) specification [32], which defines element groups and elements that describe earning resources. The IMS and ADL both use the LOM elements and structures in their specifications.
- The ADL Sharable Content Object Reference Model (SCORM) specification [ADL2006] combines elements of IEEE, AICC and IMS specifications into a consolidated document that can be "easily" implemented. The ADL adds value to existing standards by providing examples, best practices and clarifications that help suppliers and content developers implement eLearning specifications in a consistent and reusable way.

There are already a few attempts to combine the adaptive hypermedia and metadata-based courseware re-use approaches in one framework such as WHURLE [5], ALE [47] and PaKMaS [Süß2000]. AHA! uses XML-based data structures similar to many of the elearning standards and elearning systems and so, it can be easly translated to other systems and specifications. Some good examples are the Interbook to AHA! compiler [21], the integration of AHA! and Auld Linky [33], the integration of MOT with AHA! [17], a tool for importing/exporting AHA! courses to/from SCORM courses [40] and a tool [41] for importing/exporting AHA! assessments to/from other tests systems sush as IMS QTI [4], QuestionMark [38], SIETTE [3], Moodle [35], etc.

Finally, in Table 1, we show a resource list (software, conferences and journals) about Web-based adaptive educational systems to enable the reader to consult more information on these type of systems.

Table 1. Resource list about authoring Web-based adaptive educational systems.

Type	Name	Website
Software	ELM-ART	http://apsymac33.uni-trier.de:8080/Lisp-Course
	InterBook	http://www.sis.pitt.edu/~peterb/InterBook.html
	AHA!	http://aha.win.tue.nl/
	KBS Hyperbook	http://www.kbs.uni-hannover.de/hyperbook/
	SQL-Tutor	http://www.cosc.canterbury.ac.nz/tanja.mitrovic/sql-tut.html
	NetCoach	http://www.orbis.de/netcoach
	MetaLinks	http://ddc.hampshire.edu/metalinks/
	WHURLE	http://whurle.sourceforge.net/
	RATH	http://wundt.kfunigraz.ac.at/rath/
Conferences	Adaptive Hypermedia and Adaptive Web-Based Systems (AH)	http://www.ah2006.org/
	World Conference on Educational Multimedia, Hypermedia and Tele-communications (ED-Media)	http://www.aace.org/conf/edmedia/
	International Conference on Artificial Intelligence in Eduation (AI-ED)	http://hcs.science.uva.nl/AIED2005/
	ACM Conference on Hypertext and Hyper-media (Hypertext)	http://hypertext.expositus.com/
	World Conference on E-learning (E-Learn)	http://www.aace.org/conf/eLearn/default.htm
Journals	New Review of Hyper-media and Multimedia	http://www.gbhap.com/journals/titles/13614568.asp
	User Modeling and User-Adapted Interac-tion (UMUAI)	http://www.umuai.org/

3 AHA! Architecture

AHA! [19] [20] [22] [23] is a Java-servlet-based software environment that works with the Tomcat web server, on Linux (or UNIX) as well as on Microsoft Windows. It is available from http://aha.win.tue.nl/. Figure 1 shows the overall architecture of AHA! system. The core is formed by the AHA! engine which is implemented using Java Servlets running on (and communicating with) the web-server. The information on the server consists of three parts we describe in detail below: a combined domain and adaptation model (DM/AM), corresponding to these models in AHAM, a user model (UM) which keeps track of the user's knowledge about the domain concepts, and the local pages which contain the content of the application or course. It is possible to include external pages (retrieved from other web servers) and they are (potentially) adapted in the same way as local pages. AHA! also contains authoring and management tools, explained in a later section.

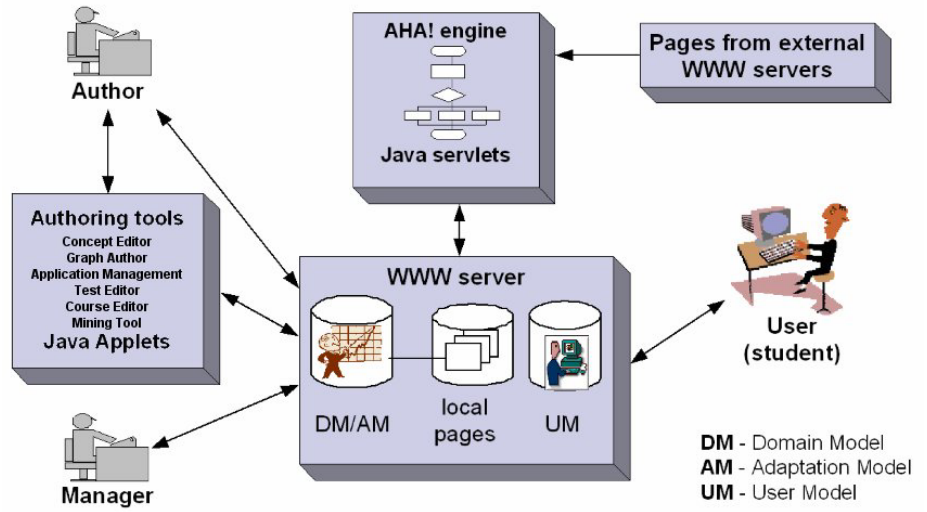

Fig. 1. Overall AHA! architecture.

AHA! is an Open Source Web server extension to add adaptation to applications such as on-line courses. Users request pages by clicking on links in a browser, and AHA! delivers the pages that correspond to these links. However, in order to generate these pages AHA! uses three types of information:

- The domain model (DM) contains a conceptual representation of the application's content. It consists of concepts and concept relationships. In AHA! every page that can be presented to the end-user must have a corresponding concept. It is also possible to have conditionally included fragments in pages. For each place where a decision needs to be made what to include a concept must be defined. (Such a concept can be shared between different pages on which the same information is conditionally included.) Pages are normally grouped into sections or chapters or other high-level structures. AHA! makes use of a concept hierarchy through which one can easily have knowledge propagated from pages to sections and chapters, and through which AHA! can automatically generate and present a hierarchical table of contents. Concepts can be connected to each other through concept relationships. In AHA! there can be arbitrarily many types of concept relationships. A number of types are predefined to get you going quickly as an author. A typical example of a (predefined) relationship type is prerequisite. When concept A is a prerequisite for concept B the end-user should be advised (or forced) to study or read about concept A before continuing with concept B. This relationship is translated into adaptation rules that will adapt the colors of anchors for links to concept B. The translation from concept relationships to the actual adaptation using colors and/or icons is very flexible but defined through templates the average author may not wish to change. The flexible scheme is explained in [23]. By default the link anchors will have the normal Web colors (blue or purple) when the link leads to a concept (or page) for which all the prerequisites are met, and will have the color black when some prerequisites are not met. Creating a domain model, using existing templates, is easiest using the Graph Author tool.
- The user model (UM) in AHA! consists of a set of concepts with attributes (and attribute values). This model contains an overlay model, which means that for every concept in DM there is a concept in UM. UM can contain additional concepts (that have no meaning in DM) and it always contains a special pseudo-concept named personal. This concept has attributes to describe the user and includes such items as login and password. When a user accesses an AHA! application the login form may contain arbitrary (usually hidden) fields that contain values for attributes of the personal concept. It is thus possible to initialize preferences through the login form. To get you going quickly as an author the AHA! authoring tools provide a number of UM concept templates, resulting in concepts with predefined attributes. Typical attributes are *knowledge* and *interest*, to indicate the user's knowledge of or interest in a certain concept. AHA! will automatically propagate an increase in knowledge of a concept to higher-level concepts (higher in the concept hierarchy of DM). It will also record a lower knowledge increase when studying concepts for which the prerequisites are not yet known.
- The adaptation model (AM) is what drives the adaptation engine. It defines how user actions are translated into user model updates and into the generation of an adapted presentation of a requested page. This model consists of adaptation rules that are actually event-condition-action rules. Most authors will never have to learn about AM because the rules are generated automatically by the Graph Author. Whe you have very specific adaptation needs (not possible with the existing templates) you should either learn to create your own templates or study the Concept

Editur that lets you create arbitrary adaptation rules. We now explain what happens exactly when the end-user clicks on a link in a page served by AHA!:

1. In AHA! there are two types of links: links to a page and links to a concept. Since in DM pages are linked to concepts AHA! can find out which concept corresponds to a page and which page corresponds to a concept.
2. The adaptation engine starts by executing the rules associated with the attribute access of the requested concept (or the concept that corresponds to the requested page). Access is a system-defined attribute that is used specifically for the purpose of starting the rule execution.
3. Each rule may update the value of some attribute(s) of some concept(s). Each such update triggers the rules associated with these attributes of these concepts.
4. When the rules have been executed AHA! determines which page was requested. (It is possible to associate several pages with a concept and have rules decided which page to present, just like with conditional fragments inside a page. Details are described in [23].)
5. A frame is generated with components that define the look and feel of the application. The presentation may include a hierarchical menu with chapters and sections for instance. One frame (typically the largest one) is used to present the requested frame.
6. The requested page is parsed and adapted. This adaptation includes changing the link colors and conditionally including fragments (also called objects). The insertion of objects actually has some complex side effects that we do not discuss further in this chapter.

In AHA! the presentation of a course can be influenced in many more ways that are beyond the scope of this chapter. AHA! uses a layout model to define how concepts (of different types) are presented. A layout is basically an HTML frames structure, and apart from a frame that shows a page there can be frames that show part of a table of contents, concepts of which the knowledge is increased by reading the current page, prerequisite concepts (and their knowledge level), etc. Through the powerful layout model AHA! applications can be made to look very similar to applications in other adaptive educational hypermedia platforms. Figure 2 shows the AHA! tutorial as an AHA! application, using just one of many ways in which the presentation is possible.

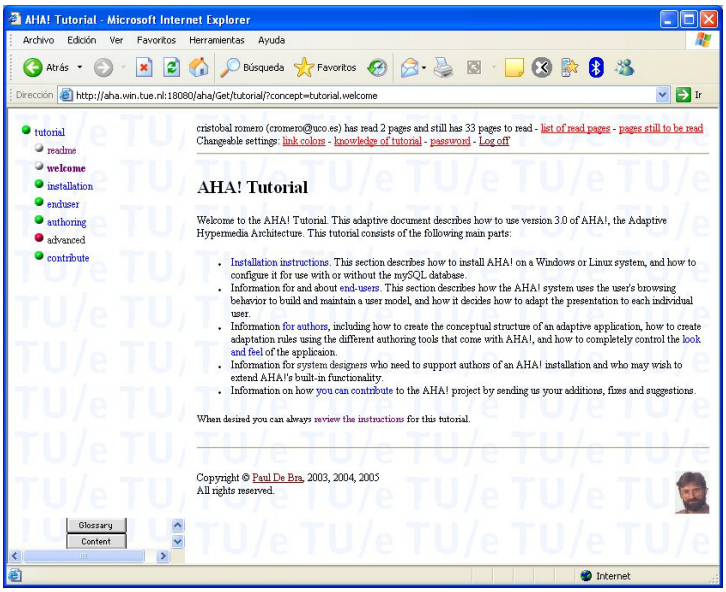

Fig. 2. AHA! Tutorial.

4 Authoring and Management AHA! Tools

Authors in AHA! are special users, created by the manager (who installs AHA!). As an author you can access the different authoring tools and your account information through the main starting page of the AHA! distribution. From this page (see Figure 3) the author can go to the following sub-parts:

- The Configuration is an interface for manipulating the different AHA! data structures. It lets you choose between the use of XML files or a mySQL database for the concept structures and the user models, and lets you convert authoring formats to internal formats. It lets you define authors and assign applications to authors.
- The Application Management Tool is the "parent" authoring tool. It lets you transfer files from your local machine to the server and back, and lets you start most other authoring tools by simply clicking on the names of the authoring files.
- The Author Workplace to change your settings as an author, and access all the authoring tools (Concept Editor, Graph Author, Form Editor, Test Editor, etc.). Authorization is required to change your author settings. You can also access the authoring tools directly from the authoring page. (The tools ask for authorization when you start them.)
- The starting page may provide links that lead to hyperdocuments or applications that are served from this instance of the AHA! system. The standard AHA! 3.0 start page offers access to one application: the AHA! 3.0 tutorial. The tutorial is

adaptive. By studying the source files for the tutorial you can also learn more about how to create adaptive documents with/for AHA!.

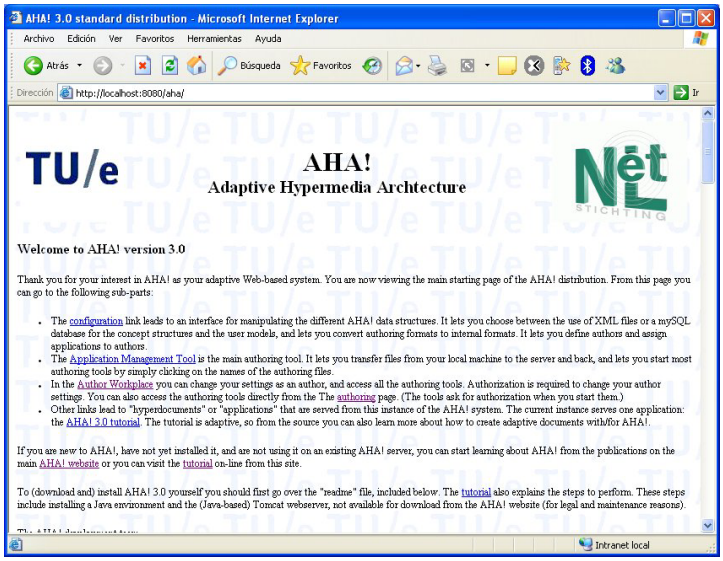

Fig. 3. Main starting page of the AHA! distribution.

4.1 Application Management Tool

The Application Management Tool [22] gives easy access to the authoring tools as well, and provides a file transfer tool that lets you copy files back and forth between your (authoring) workstation and the AHA! server. In order to manage an application as an author the application management tool, or AMT for short, offers a user-friendly interface to copy files between your local PC and the AHA! server. AMT works as a signed Java applet, in order to be able to access your local file system. The figure 4 shows the AMT interface. The left half of the window gives an explorer-like view of your local file system. The right half can either show the application files which are all the files that you create as an author and that the server uses when the application is running. The alternative view shows the author files which are the files that are created using the special AHA! authoring tools for the domain model. When you double-click on one of these files the appropriate authoring tool is started automatically. The files created by the authoring tools (Graph Author and Concept Editor) are not stored on your PC, only on the server.

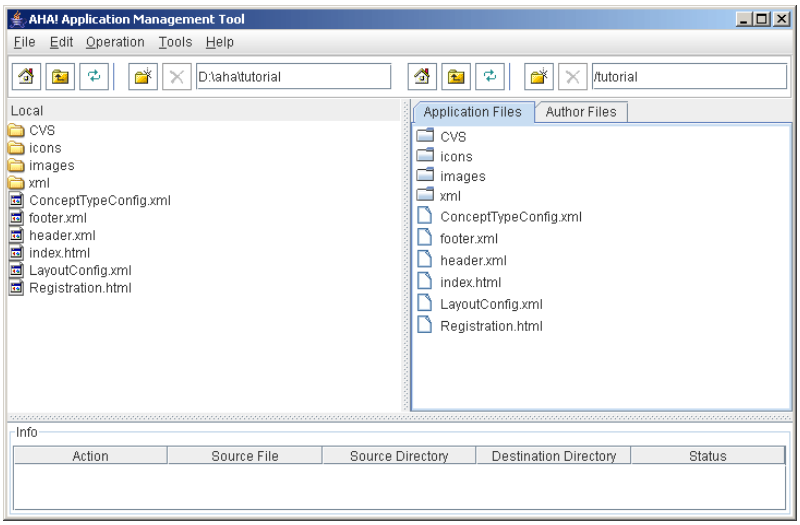

Fig. 4. Application Management Tool main window.

The AMT show an interface that is similar to that of a popular secure shell (ssh) interface. On the left you see the local file system, and on the right you either see the server's file system (in the Application Files tab) or the server-side authoring files for the Graph Author and Concept Editor. When you double-click on the .gaf files the Graph Author is started automatically and when you double-click on the .aha files the Concept Editor is started. It is thus possible to perform all the application creation and maintenance using AMT. The one part that is not supported by special tools in AHA! is the creation of the application's files like pages and images. Although AHA! contains some limited backward compatibility support for plain HTML, it works best with XHTML, with or without some AHA! extensions.

4.2 Graph Author Tool

The Graph Author [22] is the main authoring tool for the concept structure. It lets you create a concept hierarchy and concept relationships of different types. 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, based on the templates. Figure 5 shows a screenshot of the Graph Author.

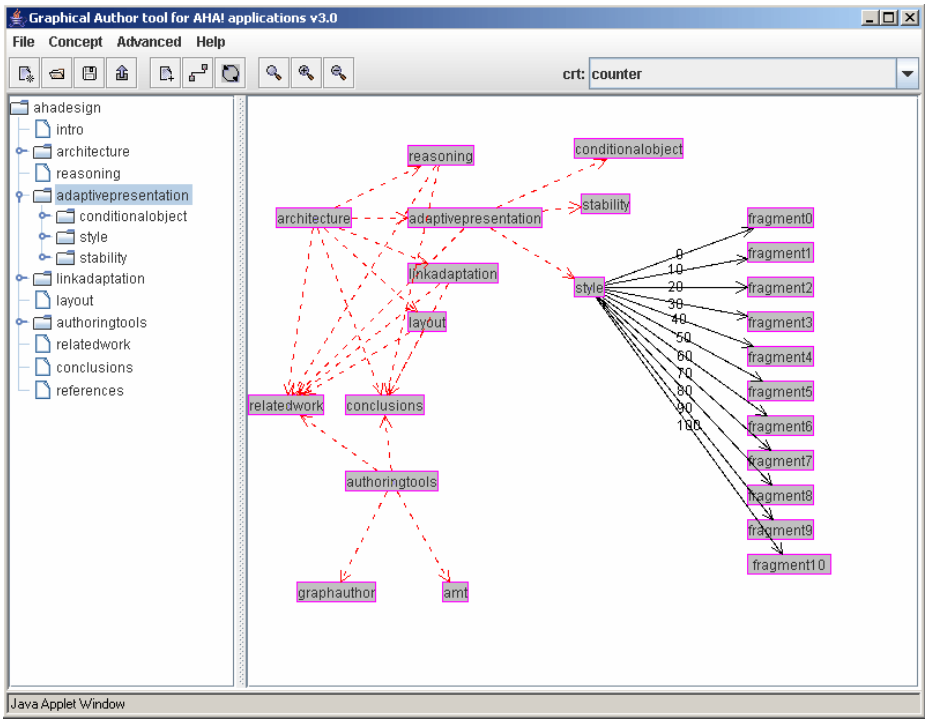

Fig. 5. Graph Author tool main window.

In the Graph Author you create concepts and concept relationships. The Graph Author window is split into two parts (see Figure 5), showing the concepts (as a hierarchy) on the left and showing the concept relationships (as a graph) on the right. The graph represents the structure of prerequisite relationships in a tutorial application. The concepts are structured as a hierarchy which in fact also is a structure of concept relationships (and always present in an AHA! application). Every type of relationship has a different meaning related to the adaptation an application provides (we explain this later) and is represented using different color and style of arrows.

The concept relationship graph is created by dragging concepts from the hierarchy shown on the left to the drawing pane, and by then drawing arrows between the concepts. You first select the appropriate concept relationship type from the drop-down list (top right in Figure 5) and then click on the source concept and drag to the destination concept. Some concept relationships may have an optional parameter. By clicking on the arrow a textfield appears in which the parameter value can be entered. For a prerequisite for instance the amount of knowledge that must be exceeded in order for AHA! to consider the prerequisite to be fulfilled is a parameter. (Its default value is 50 in this case.). Also, the rules are executed (conditionally) when a page (associated with the concept) is accessed. In an application like the tutorial one we have three types of concept relationships that play a role:

- For every page there is a unary relationship (a relationship from the page to itself), called knowledge update. When a page is read the action that is performed depends

on the suitability of the page. If the suitability attribute is true then the knowledge of the page is set to 100. If it is false then the knowledge of the page is increased to 35. (By this we mean the value is set to 35 if it is lower but left at its previous value if that was already over 35.).

- The concept hierarchy shown in the Graph Author is used for knowledge propagation. When the knowledge of a concept changes that change is propagated to the concepts that are higher in the concept hierarchy. How much knowledge is propagated depends on the number of siblings the concept has. The idea is that when all siblings reach a knowledge level of 100 the parent should have 100 as well. (But due to integer arithmetic and truncation that value may end up being slightly lower.).
- The prerequisite relationships determine the suitability of a concept. If A is a prerequisite for B, expressed by drawing a prerequisite arc from A to B in the graph, the suitability of B depends on the knowledge of A. The standard rule requires the knowledge of A to be higher than 50 in order for B to be considered suitable.

4.3 Concept Editor

The Concept Editor [22] is an authoring tool for defining concepts and adaptation rules. This tool 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. Note that whereas the Graph Author can generate files in the Concept Editor's authoring format it cannot import them. Some user interface differences between the Graph Author and the Concept Editor exist for historical reasons only. Figure 6 shows the Concept Editor with the same tutorial example used earlier.

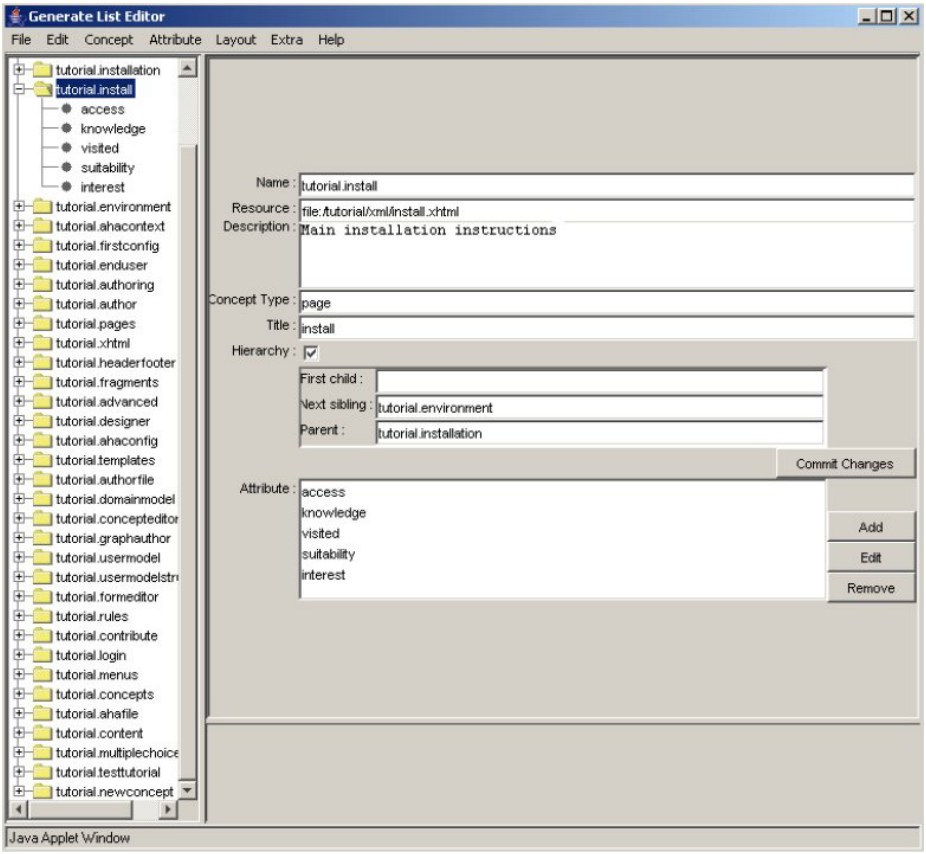

Fig. 6. Concept Editor main window.

The concept hierarchy is represented inside the concepts, but the Concept Editor (unfortunately) does not show that hierarchy in its left frame. Also, the Concept Editor is sometimes referred to as Generatelist Editor for historical reasons. The authoring format with concepts and adaptation rules is called the generatelist format. The editor lists all the concepts (of a single application) on the left, and shows details of a selected concept on the right.

4.4 Form Editor

The Form Editor [22] enables you to create custom forms to let end-users change values of attributes of concepts in their user model. Attributes of concepts defined as changeable can be included in a custom-made form. The Form Editor lets you create an (X)HTML form in which form elements for the attributes of concepts are inserted automatically, and in which you create the remainder of the presentation by means of plain HTML code. A form is bound to an application, so when creating a new form you have to load the conceptual structure of that application. (File, Load AHA! appli-

cation). The form editor creates a skeleton representing an empty form. You can add HTML code for the presentation and use the buttons Input, Select, Option and Button to add form elements. A form can be viewed as HTML source and can be previewed. The Form Editor uses a standard Java HTML editor class to do this (see Figure 7). At the time of writing this tutorial this standard class is not yet fully XHTML compliant, so you will have to use a somewhat simplified HTML, which is normally enough for a simple form. Forms created with the Form Editor are saved in your authoring directory. You have to copy them to whichever location on the server you want to use to refer to them from within the content pages that have a link to them. (We are working on a tool that lets you create forms inside the application's document tree right away.)

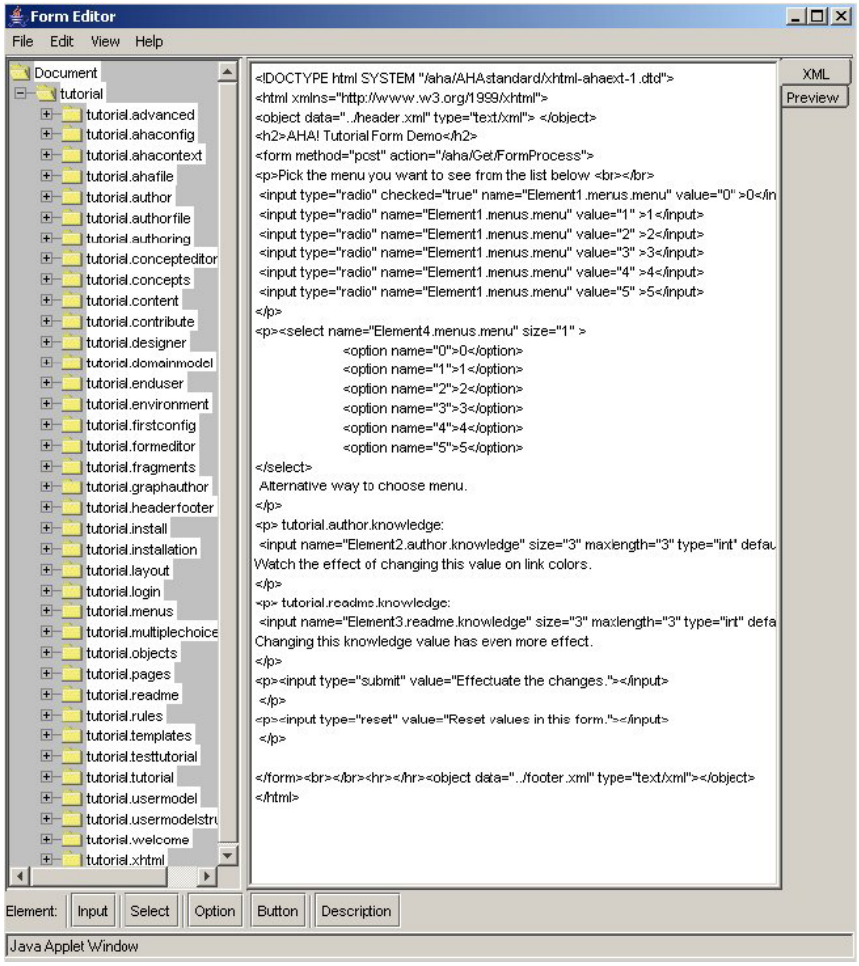

Fig. 7. Form Editor with a example form.

4.5 Test Editor

AHA! 3.0 comes with a (new) authoring tool (Test Editor) for developing multiple-choice tests that can be used in web-based systems and wireless devices [39] [41]. Computerized tests or quizzes are among the most widely used and well-developed tools in web-based education [9]. There are different types of computerized tests, depending on the type of items or questions (yes/no questions, multiple-choice/single-answer questions, fill-in questions, etc.) and there are two main types of control algorithms: classic or linear tests and adaptive tests [50]. Test Editor is an authoring tool for building adaptive (and randomized) and classic (non-randomized) multiple-choice tests. The specific life cycle of tests we have used is:

- As the first step for developing a test with Test Editor, the examiner has to create one or several (XML) *items* files. An *item* consists of a single question about a single concept (from an AHA! application or course), the answers (right or wrong) and explanations for the wrong answers. Several items/questions about the same concept can be grouped together into one items file. Figure 8 shows how to add questions to the items file, one by one. The examiner must also specify some required parameters (the enunciate flag, and for each answer a flag to indicate whether the answer is correct) and can add some optional parameters (an illustrative image, explanations and Item Response Theory (IRT) parameters [50]: item difficulty, discrimination and guessing). Using the Test Editor items can be added, modified or deleted. They can be imported/exported to/from other tests systems (IMS QTI [4], QuestionMark [38], SIETTE [3], Moodle [35] etc.). Questions can thus be re-used from other test environments without needing to enter them again.

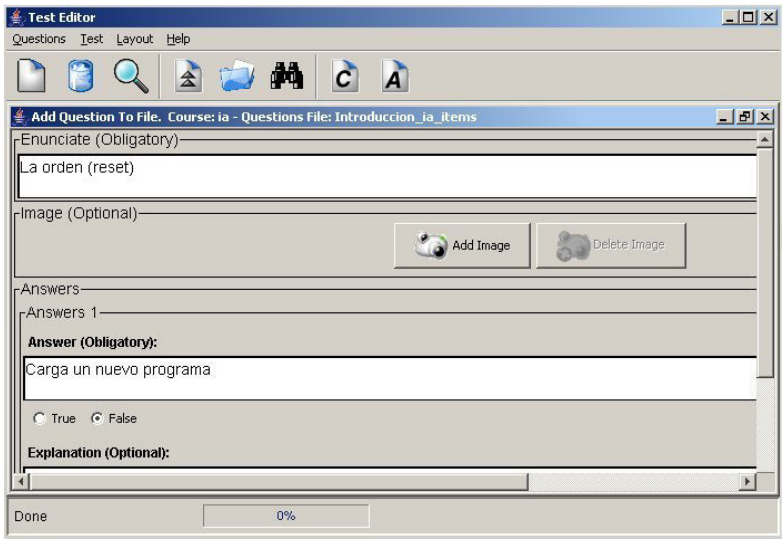

Fig. 8. Test Editor: Windows to introduce the obligatory parameters of an item.

- The second step is to build tests out of items. The examiner decides on the test type (classic test or adaptive test) he wants and whether to use just one or several items files. If the test evaluates only one concept, we consider it to be an activity. If the test evaluates several concepts, it will be an exam, about a chapter or perhaps a whole course. Next, the examiner can use different methods to select what specific items from these items files will be used in the test (the selection can be done manually, randomly or randomly with some restrictions). Then he sets presentation parameters (see Figure 9) about how questions are shown to examinees: the order in which questions and answers are shown, whether to show or hide explanations of the answers (through the verbose flag), the maximum time to respond, whether to show the correct answer or just a score, etc. In addition to these there are also parameters about evaluation: to penalize incorrect answers, to penalize unanswered questions and what percentage of knowledge the final score represents in the associated concept/concepts. If the test is adaptive, the examiner also has to set the adaptive algorithm parameters (questions selection procedure and termination criterion). Each test is stored in an XML file and that is exactly the same for both versions (PC and mobile). But for the mobile devices it also is necessary to create a *.jar* and *.jad* file that includes both the multiple-choice test code (a Java Midlet test engine) as well as the questions and parameters (XML file).

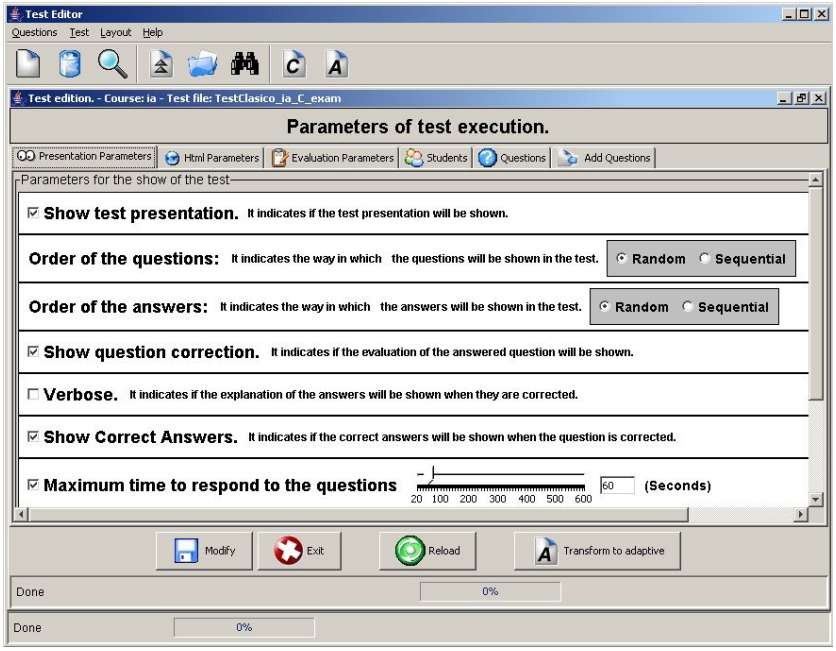

Fig. 9. Test Editor: Windows to select the questions presentation parameters.

- The generated test can be downloaded (the *.jar* file) into a mobile phone and/or can be used directly (through a browser) in an AHA! course. When used with AHA! a test is presented in an Java Applet, with a look and feel that is similar to

the Java Midlet version (see Figure 10). The results of tests are logged on the server. After a large number of examinees performed some tests, examiners can examine statistical information in the Test Editor (success rate per question, mean times to answer the questions, questions usage percentage, etc.) and use that information for maintenance and improvements of the tests. The examiner may decide to modify or delete bad items, add new items, but he can also modify the test configuration. Test Editor also can do items calibration, in order to transform a classic test into an adaptive one, or to optimize the IRT parameter of an adaptive test.

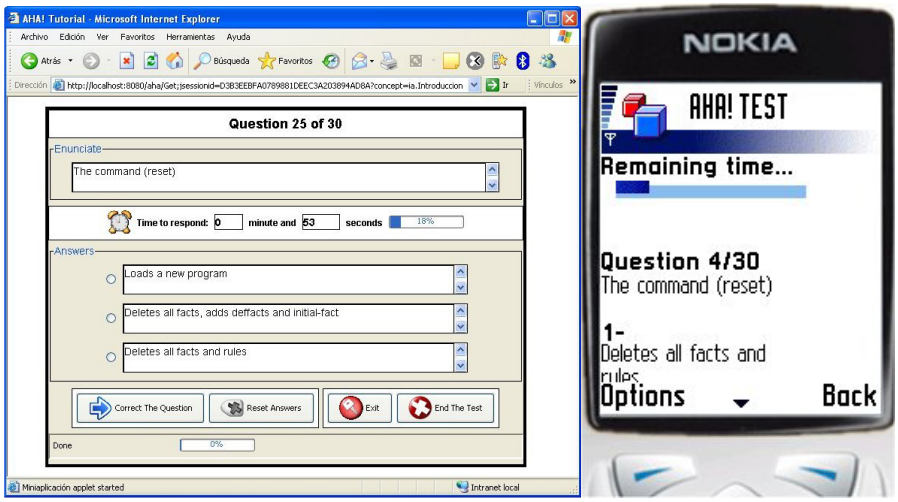

Fig. 10. Interface of a question in an AHA! course and in a mobile phone.

4.6 Mining Tool

Currently, we are developing a new mining tool in order to help authors in discovering interesting information from students' usage information that can be used to improve the courses. Currently, we have developed a "links recommendation" facility based on sequential pattern mining. The recommendation of links (to content pages, activities, etc) is very important in e-learning systems in order to personalize (or adapt) the learning for each student and to guide them to the best learning path. One way to automate this process is the application of data mining techniques into the students' usage information. In most e-learning systems, all the pages accessed by students are saved in log files (either one log file for each student or just one big log file for everyone) that they contain all the information about the interaction of the students with the system. Therefore, after pre-processing this information, it is possible to discover sequential patterns from these log files by using some data mining algorithms. Sequential pattern mining can be defined as the process of discovering all sub-sequences that appear frequently on a given sequence database and have mini-

mum support threshold. Our objective is to use the discovered sequential patterns to create interesting recommendation links to show to the students while they use the e-learning system. To do that, all the sequential patterns are split in sequences of only two components. These obtained sequences can be considered as a rule with only one antecedent and one consequent, so that the antecedent represents the page in which the recommendation is shown and the consequent is the link recommended to the student.

We have developed a mining tool (see Figure 11) in order to help to the teacher to carry out all this process. This application is a Java Applet, just like other AHA! authoring tools. In order to use it, the author has to identify himself and to choose from which courses and students he wants to discover sequential patterns. Then, the application creates a file with the pre-processed data in Weka format [53]. Next, the author has to select a data mining algorithm for extracting sequential patterns from the available ones [2]: AprioriAll, GSP and PrefixSpan. When the algorithm finishes its execution, the discovered sequences are shown and they can be automatically translated into recommendations links to be inserted into the corresponding course web page.

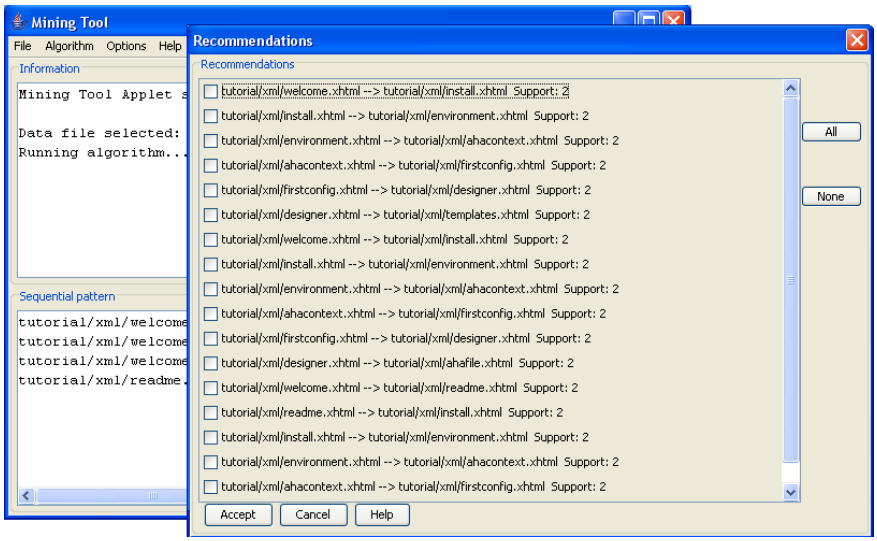

Fig. 11. Recommendation links window and application main window.

4.7 Course Editor

At the University of Córdoba we are also working on a new high level tool, named Course Editor (see Figure 12). Using this tool, (of which the completion date is currently still undetermined, authors can easily create and maintain AHA! courses. This tool can create a new course or import and export AHA! courses to/from SCORM courses [40], can edit the XHTML course pages, can visually configure the layout of

the course, can add collaborative services like a chat, an announcement board, an upload tool, etc. to a course.

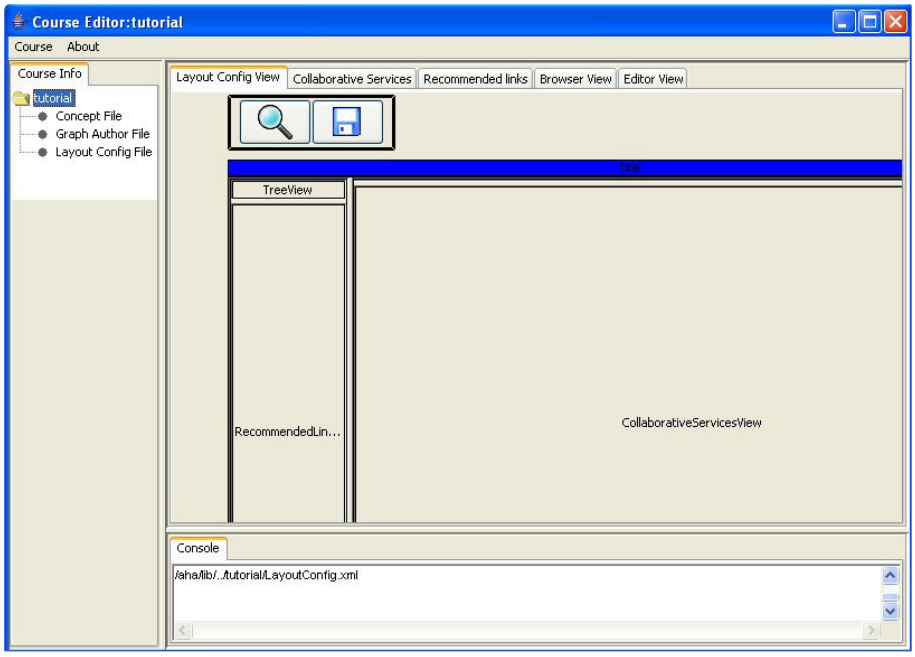

Fig. 12. Course editor main window.

The mains functions that the Course Editor tool provides to the author are:

- **Creating new courses.** Course Editor can create a new complete AHA! course. It creates not only the configuration files (.aha and .gaf) but also all the directories and pages (.xml, .html, etc.) with the content of the course. It can also open a previously created AHA! course in order to edit and add new characteristics.
- **SCORM Export and Import.** Course Editor provides a way to create a new AHA! course from a SCORM 2004 package course [43]. So, it's very easy to create a course or a base-plane for a new course. It also provides a way to export AHA! courses to SCORM courses in order to reuse courses previously created in other e-learning environments. (Conversion to SCORM may be lossy, depending on which adaptation possibilities AHA! offers are used.)
- **Layout View configuration.** With Course Editor, an author is able to change the layout of the course, change sizes of frames in a WYSIWYG way or add/remove views (corresponding to HTML frames) to the main window.
- **Collaborative Services.** Course Editor allows author to include a new layout in the main window (or in a secondary one), giving access to collaboration tools. Adding collaborative services such as Chat, Announcement Board, Upload and Download, enables students to interact with the teacher and with others students using the same platform, or even the same course.

- **Recommended links.** With these recommended links, authors can include direct links to other pages. These recommendations are a type of relationship between concepts. The teacher can use his personal experience, history of students' results, a data mining study, etc. to establish them. These direct relationships are shown in the student's browser window, as direct links between concepts related to each other, external links or links to a test where AHA! may evaluate the student's progress and understanding of the concept.
- **Navigation Panel.** With this panel, authors are able to see the all the pages and concepts that a course has, as well as the relationships between these concepts.
- **Complete XHTML Editor.** Even if authors have no knowledge of XHTML, sometimes it is needed to edit these files. For example, to create or modify a course web page to include special AHA! tags, for a conditionally included fragment or for AHA!-specific anchors. Editing will be performed by using a WYSIWYG tool, which will include AHA! options as additional option to standard XHTML editing.
- **Browser View.** Course Editor provides a way to view a page of the course as it may appear when students will see it. (We say "may" because the actual presentation depends on the student's user model.) This view will be shown in an embedded browser in the Course Editor, Internet Explorer or whatever browser is the default on the author's machine.

In the Figure 13 you can see the interface of a course edited with Course Editor in which we have added some Collaborative Services and Recommended Links; in this case, the student is recommended to go to the "graph author" concept if he is currently reading the "pages" concept.

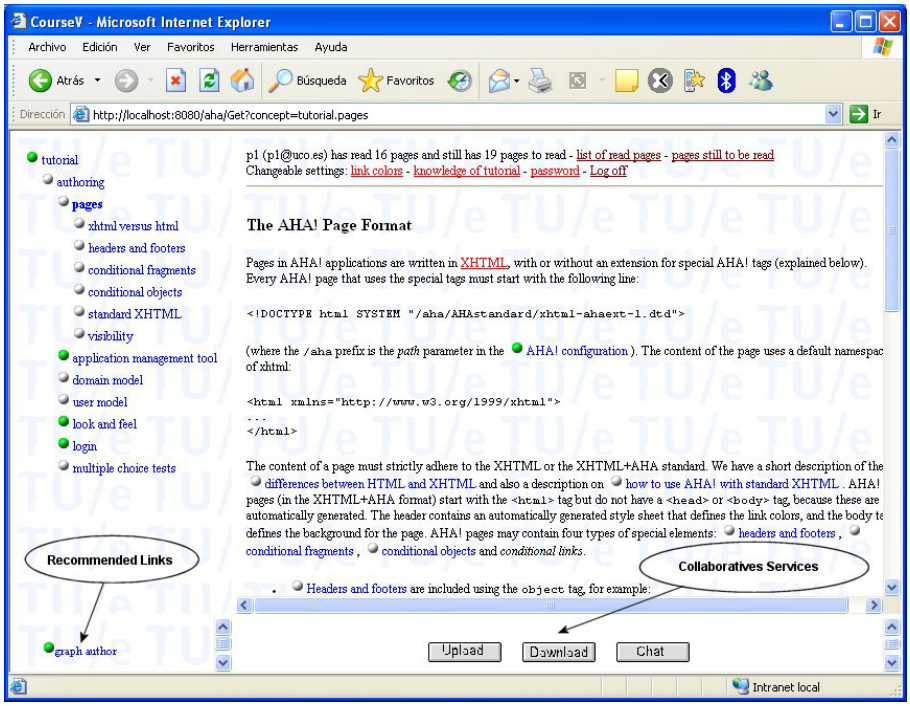

Fig. 13. AHA! Tutorial with Collaborative Services and Recommended Links.

5 Conclusion

In this chapter we have described some authoring and management tools of the AHA! system. AHA! is one of the first and most extended adaptive systems in the world. The AHA! project, which stands for Adaptive Hypermedia for All builds on the Open Source AHA! system (the Adaptive Hypermedia Architecture), being developed a the Eindhoven University of Technology, in the Database and Hypermedia group. We have described the general architecture and the specific functionality of the AHA! system. We have described some of the mains tools provided by the currently distributed AHA! version such as: Concept Editor, Graph Author, Application Management, Form Editor and Test Editor. We described some new tools that we are developing such as: Course Editor and Mining Tools. Using all these tools the development and maintenance of AHA! applications is made easier for non-technical authors. Some of these tool are oriented to adaptive hypermedia applications in general (Concept Editor, Graph Editor, Application Management) and others are more oriented to educational applications (Test Editor, Course Editor, Mining Editor).

Currently we are developing the Course Editor and we want to improve the Mining tool in order to add to it more data mining methods as classification, clustering, association, prediction, etc. in order to enable the discovery of much more interesting information to teacher. We are also working on a tool for visualizing the student's usage data. Using this tool the teacher can see graphically all the usage information of a whole class or of specific students (visited pages, access or reading times, obtained scores, etc.).

Finally, AHA! is continuously being further developed and improved. We welcome contributions from other groups and will do our best to include the contributions in future releases.

References

1. AICC: http://www.aicc.org (2006)
2. Antunes, C., Oliveira, A.L.: Generalization of Pattern-Growth Methods for Sequential Pattern Mining with Gap Constraints. Machine Learning and Data Mining in Pattern Recognition (2003) 239-251
3. Arroyo, I., Conejo, R., Guzman, E., Wolf, B.P.: An Adaptive Web-based Component for Cognitive Ability Estimation. Proc. of Artificial Intelligence in Education. Amsterdam:IOS (2001) 456-466
4. Bacon, D.: IMS Question and Test Interoperability. MSOR Connections, 3:3 (2003) 44-45
5. Brailsford, T. J., Stewart, C. D., Zakaria, M. R., Moore, A.: Autonavigation, links, and narrative in an adaptive Web-based integrated learning environment. Proceedings of World

Wide Web Conference (2002)
6. Brusilovsky, P., Schwarz, E., Weber, G.: ELM-ART: An intelligent tutoring system on World Wide Web. Third International Conference on Intelligent Tutoring Systems (1995) 261-269.
7. Brusilovsky, P.: Methods and techniques of adaptive hypermedia. User Modeling and User-Adapted Interaction, Vol. 6:2-3 (1996), 87-129
8. Brusilovsky, P., Eklund J., Schwarz E.: Web-based education for all: A tool for developing adaptive courseware. Computer Networks and ISDN Systems, Vol. 30:1-7 (1998) 291-300.
9. Brusilovsky, P., Miller P.: Web-based Testing for Distance Education. Proc. of the World Conference of WWW and Internet, (1998) 149-154
10. Brusilovsky, P.: Adaptive Educational Hypermedia. Proceeding of Tenth International PEG Conference, (2001) 8-12
11. Brusilovsky, P.: Developing adaptive educational hypermedia systems: From design models to authoring tools. In: T. Murray, S. Blessing and S. Ainsworth (eds.): Authoring Tools for Advanced Technology Learning Environment. Kluwer Academic Publishers (2003) 377-409
12. Brusilovsky, P., Peylo, C.: Adaptive and Intelligent Web-based Educational Systems, International Journal of Artificial Intelligence in Education, Vol. 13 (2003) 156-169
13. Carmona, C., Bueno, D., EduardoGuzman, Conejo, R.: SIGUE: Making Web Courses Adaptive. Proceedings of Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (2002) 376-379
14. Carro, R. M., Pulido, E., Rodrígues, P.: TANGOW: Taskbased Adaptive learNer Guidance on the WWW. Computer Science Report. Eindhoven University of Technology (1999) 49-57
15. Cristea, A.I., De Bra, P.: ODL Education Environments based on Adaptability and Adaptivity, Proceedings of the AACE E-Learn'2002 conference (2002) 232-239.
16. Cristea, A.: Adaptive Patterns in Authoring of Educational Adaptive Hypermedia. International Peer-Reviewed On-line Journal Educational Technology and Society, Vol. 6:4 (2003) 1-5.
17. Cristea, A.I., Floes, D., Stach, N., De Bra, P.: MOT meets AHA!. Proceedings of the PEG Conference (2003)
18. De Bra, P.: Teaching Hypertext and Hypermedia through the Web. Journal of Universal Computer Science, Vol. 2:12 (1996) 797-804
19. De Bra, P., Calvi, L.: AHA! An open Adaptive Hypermedia Architecture. The New Review of Hypermedia and Multimedia, Vol 4 (1998) 115-139
20. 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 (2003) 81-84
21. De Bra, P., Santic, T., Brusilovsky, P.: AHA! meets Interbook, and more... Proceedings of the AACE ELearn (2003) 57-64
22. De Bra, P., Stash, N., Smits, D.: Creating Adaptive Web-Based Applications, Tutorial at the 10th International Conference on User Modeling (2005) 1-33.
23. De Bra, P., Smits, D., Stash, N., Creating and Delivering Adaptive Courses with AHA!, Proceedings of the first European Conference on Technology Enhanced Learning (2006) 21-33
24. Eliot, C., Neiman, D., Lamar, M.: Medtec: A Web-based intelligent tutor for basic anatomy. Proceedings of World Conference of the WWW, Internet and Intranet (1997) 161-165.

25. Gilbert, J. E., Han, C. Y.: Arthur: Adapting Instruction to Accommodate Learning Style. Proceedings of World Conference of the WWW and Internet (1999) 433-438.
26. Henze, N., Naceur, K., Nejdl, W., & Wolpers, M. : Adaptive hyperbooks for constructivist teaching. Künstliche Intelligenz, Vol. 4 (1999) 26-31.
27. Hockemeyer, C., Held, T., Albert, D.: RATH - A relational adaptive tutoring hypertext WWWenvironment based on knowledge space theory. Proceedings of International conference on Computer Aided Learning and Instruction in Science and Engineering (1998) 417-423.
28. IMS: http://www.imsglobal.org (2006)
29. Kay, J., Kummerfeld, B.: User models for customized hypertext. In C. Nicholas, J. Mayfield (eds.): Intelligent hypertext: Advanced techniques for the World Wide Web (1997)
30. Kayama, M., Okamoto, T.: A mechanism for knowledge-navigation in hyperspace with neural networks to support exploring activities. Proceedings of Workshop Current Trends and Applications of Artificial Intelligence in Education at the 4th World Congress on Expert Systems (1998) 41-48.
31. Laroussi, M., Benahmed, M..: Providing an adaptive learning through the Web case of CAMELEON. International conference on Computer Aided and Instruction in Science and Engineering (1998) 411-416
32. LOM: http:// www.ltsc.ieee.org (2006)
33. 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 (2003) 161-169
34. Mitsuhara, H., Kurose, Y., Ochi, Y., Yano, Y.: ITMS: Individualized Teaching Material System-adaptive integration of web pages distributed in some servers. Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications, (2001) 1333-1338
35. Moodle: http://moodle.com (2006)
36. Murray, T.: MetaLinks: Authoring and affordances for conceptual narrative flow in adaptive hyperbooks. International Journal of Artificial Intelligence in Education, Vol. 13:2-4 (2003) 197-231
37. Neumann, G., Zirvas, J.: SKILL - A scallable internet-based teaching and learning system. Proceedings of World Conference of the WWW, Internet, and Intranet (1998) 688-693
38. QuestionMark: http://www.questionmark.com (2006)
39. Romero, C., De Bra, P., Palomo, S., Ventura, S.,: An Authoring tool for web-based adaptive and classic tests.World Conf. on E-learning in Corporate, Government, Healthcares & Higher Education, (2004) 174-177
40. Romero, C., Rider, J., Ventura, S., Hervás, C.: AHA! meets SCORM. IADIS ELearn. Virtual Multiconference on Computer Science and Information Systems (2005).
41. Romero, C., Ventura, S., Hervás, C. de Bra, P.: An Authoring Tool for Building Both Mobile Adaptable Tests and Web-Based Adaptive or Classic Tests. Int. Conference Adaptive Hipermedia (2006) 203-212
42. Sanrach, C., Grandbastien, M.: ECSAIWeb: A Web-based authoring system to create adaptive learning systems. Adaptive Hypermedia and Adaptive Web-based Systems (2000) 214-226
43. SCORM: http://www.adlnet.gov/scorm (2006)
44. Schöch, V., Specht, M., Weber, G.: "ADI" - an empirical evaluation of a tutorial agent. Proceedings of World Conference on Educational Multimedia and Hypermedia and World

Conference on Educational Telecommunications (1998) 1242-1247
45. Specht, M., Weber, G., Heitmeyer, S., Schöch, V.: AST: Adaptive WWW-Courseware for Statistics. Proceedings of Workshop Adaptive Systems and User Modeling on the World Wide Web at 6th International Conference on User Modeling (1997) 91-95.
46. Specht, M., Oppermann, R.: ACE - Adaptive Courseware Environment. The New Review of Hypermedia and Multimedia, Vol. 4 (1998) 141-161.
47. Specht, M., Kravcik, M., Klemke, R., Pesin, L., Hüttenhain, R.: Adaptive Learning Environment (ALE) for Teaching and Learning in WINDS. International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (2002) 572-581
48. Vassileva, J., Deters, R.: Dynamic courseware generation on the WWW. British Journal of Educational Technology, Vol. 29:1 (1998) 5-14
49. Vassileva, J.: DCG + GTE: Dynamic Courseware Generation with Teaching Expertise. Instructional Science, Vol. 26:3-4 (1998) 317-332
50. Wainer, H.: Computerized Adaptive Testing: A premier. New Jersey, Lawrence Erlbaum Associates (2000)
51. Weber, G.: ART-WEB. Trier Ed.: University of Trier (1999)
52. Weber, G., Kuhl, H.-C., Weibelzahl, S.: Developing adaptive internet based courses with the authoring system NetCoach. Proceedings of Third workshop on Adaptive Hypertext and Hypermedia (2001) 35-48
53. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann (2005)