Extending AHA!

Cristóbal Romero Morales

Sebastián Ventura Soto

Cesar Hervás Martínez

*Paul de Bra

Departament of Computer Science

University of Cordoba

14071 Campus de Rabanales,

Cordoba, SPAIN

Tlf:34957218630, Fax:34957218630

cromero@uco.es, sventura@uco.es, chervas@uco.es

*Eindhoven University of Technology

Department of Computer Science

PO Box 513, NL 5600 MB Eindhoven

The Netherlands

debra@win.tue.nl

Extending AHA!

Adding Levels, Data Mining, Tests and SCORM to AHA!

Abstract: In this paper we are going to show some specific extensions of the AHA! system developed at Córdoba University. AHA! is an open source general-purpose adaptive hypermedia system. We have developed some extensions in two different AHA! versions in order to increase its adaptation power in e-learning. In AHA! version 1.0 we added facilities for managing different students' knowledge levels and we developed a specific data mining tool. In the current AHA! version 3.0 we have developed an authoring tool to create both classic and adaptive computerized test and we have developed a tool to import SCORM content.

Keywords: AHA!, data mining, computerized test, SCORM.

1 Introduction

In the past years, we have seen an exponential growth in the use of web-based technology in distance learning systems. At the same time, more and more artificial intelligence techniques have been applied to these systems to improve students' learning, named as 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 (Brusilovsky, 2003). Adaptive educational hypermedia 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. There are several modern authoring systems to develop adaptive educational hypermedia (Murray, 2003): InterBook, Web DCG, AHA!, ACE, ALE, NetCoach, ECSAIWeb, MetaLinks, etc., all oriented to educational practitioners.

The AHA! system (De Bra, 1998) (De Bra, 2003), or Adaptive Hypermedia Architecture, was designed and implemented at the Eindhoven University of Technology, and sponsored by the NLnet Foundation through the AHA! project, or Adaptive Hypermedia for All. 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. We have chosen AHA! because is an open source general-purpose adaptive hypermedia system and it is an extended web-based platform which can provide adaptive e-learning. 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, Test Editor, etc.) to allow authors to easily create or change applications or courses, concepts, concept relationships, computerized tests, etc.

In this paper, we first describe the overall AHA! architecture and then discuss the new extensions we developed. We conclude by showing ongoing and future research and developments.

2 AHA! Architecture

AHA! (De Bra, 1998) (De Bra, 2003) is an open source 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/. AHA! works as a Web server. 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. In AHA! prerequisite relationships result in changes in the presentation of hypertext link anchors. By default the link anchors will have the normal Web colors (blue or purple) when the link leads to a page concept for which all the prerequisites are met, and will have the color black when some prerequisites are not met. Creating a domain model is easiest using the Graph Author tool.

- The *user model* (UM) in AHA! consists of a set of *concepts* with *attributes* (and *attribute values*). UM contains an *overlay model*, which means that for every concept in DM there is a concept in UM. In addition to this, 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. AM 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, but in order to really create the adaptive applications that do exactly what you want you should get to know either the *Concept Editor* presented in Sect. 6 or get to know how to define *concept relationship templates* used by the Graph Author. Now that we have seen the "components" that make up an AHA! application we can explain what exactly happens when the end-user clicks on a link in a page served by AHA!:

    - 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.

    - 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.

    - 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..

    - 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 decide which page to present, just like with conditional fragments inside a page.)

    - 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.

    - 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 paper.

[Insert Figure 1]

2 Knowledge Levels in AHA!

We have seen that AHA! is a general architecture for building adaptive hypermedia applications. It allows for fine-grained adaptation based on the knowledge of individual page-concepts, but also for coarse-grained adaptation based on the knowledge of whole chapters. In (Romero et al., 2002) an earlier version

of AHA! (version 1.0) was extended with functionality for adaptation to the student's *knowledge level* of a whole chapter. (In AHA! 3.0 no extension would be needed to realize this functionality). The approach taken was to provide learning material at three levels of difficulty, and to guide the students through these levels for each chapter.

- Domain Model. One course consists of several chapters with several concepts, but the concepts and the questions related with these concepts are divided in three levels of difficulty (high, medium or low).

- User Model. The student's knowledge for each concept, initial test or final test can be only one of these values: 0 (not yet read), 10 (low), 50 (medium) and 100 (high).

- Adaptation engine. Before studying a new chapter the students have to do an initial adaptive test to discover their initial knowledge level. The system then presents them only the concepts with this level. Each concept has an activity to evaluate the student's knowledge about this specific concept. When the students have visited all the concepts they have to do a final (multiple-choice) test to evaluate the their knowledge about the chapter *at this level*. If they obtain a medium or high level in the final test they can go to a higher level. If they succeed in the test at the highest level they can go to the next chapter. In each chapter everything starts again (see Figure 2): initial test to determine the entry level, studying pages, doing activities and taking the final test at the entry level, then at the next level, etc., until the chapter is finished at the highest level.

[Insert Figure 2]

Based on this modified AHA! version, we developed a course about the operating system Linux. The combination of initial tests and adaptation at different knowledge levels made it possible for students to skip parts of the course at levels below their prior knowledge. This is especially useful for a course (like Linux) that is given to some students who have absolutely no knowledge of the topic and some students who have some knowledge of some parts of the course already.

In figure 3, the same chapter from the Linux course is shown at three different difficulty levels: BEGINNER (left), NORMAL (middle) and EXPERT (right). Each version starts with a different concept explanation that is suited to the corresponding knowledge level. This presumed knowledge level is also expressed by the background color of the pages and the text label: Nivel 0 (BEGINNER), Nivel 1 (NORMAL) and Nivel 2 (EXPERT).

[Insert Figure 3]

3 Adding a specific data mining tool

The application of knowledge discovery techniques and data mining in web-based education systems is a very novel and promising research area (Zaïene, 2001). The same idea has been successfully used for a long time in e-commerce systems (Spiliopoulou, 2000). But whereas the e-commerce objective is to guide clients in making purchase decisions, the e-learning objective is to guide students in learning. Currently there are a lot of tools, both commercial and freeware, to carry out data mining tasks, and mainly rule discovery. Among all those, DBMiner (Zytkow & Klosgen, 2001) and Weka (Witten & Frank, 1999) stand out because they are very popular public domain systems and they have an integrated graphical environment that lets them carry out almost all data mining tasks. The main inconvenience is that those tools are difficult to use for a non-expert in data mining. In addition they are of a general purpose nature, so they can't carry out a specific treatment of domain knowledge. In particular they do

not contain features that are specific to the area of Adaptive Systems for Web-based Education (ASWEs) (Brusilovsky, 2003). To resolve these problems we have developed a specific tool, denominated EPRules (Education Prediction Rules), to simplify the process of discovering prediction rules (Romero et al., 2003) from AHA! (version 1.0) usage information.

EPRules is a visual tool to discover prediction rules and it is oriented to be used by the teacher or author of the course. It has been implemented in the Java programming language and its main feature is its specialization in education through attributes, filters and specific restrictions for the ASWE domain. Furthermore, it is a dynamic tool, because it lets the (advanced) user add new rule discovery algorithms and new rule evaluation measures, by modifying only some configuration files (Java properties files) and by selecting the new types in the algorithm directory or the new evaluation measures. The graphical interface of the EPRules application consists of four main windows:

- Data input: In this window you can either open an existing database with the usage data of an AHA! course or create a new database and add new students to it. To create a new database or to add data, the course usage files must be selected (students' log files) that will be pre-processed and integrated into a relational database. The algorithm parameters of the time-variable discretization can be also selected and configured. (This consists of transforming continuous attributes into discrete attributes). The following unsupervised global methods have been implemented: the equal-width method, the equal-frequency method and the manual method (in which you have to specify the cut-off points.). We have only discretized the time attribute, assigning three values: HIGH, MEDIUM and LOW.
- Data view: This window shows the students' pre-processed usage data. These data are about the access times, correct answers and knowledge levels obtained by students for the different web pages (activities and contents) that make up the course. You can select either visualizing all students' data, an specific student's data, a particular theme of the course, a specific concept of the theme, the visibility and difficulty level of a particular theme (high, normal, low), or a type of particular information (time, level or correct answers).
- Rule discovery: This is the most important part of the tool because this is where the different algorithms for rule discovery are applied. The implemented algorithms are algorithms for decision tree building like the ID3 (Quilan, 1987), association rules algorithms (Agrawal et al., 1973), an inductive learning algorithms (Cendrowska, 1987) and different versions of evolutionary algorithms, specifically the grammar based genetic programming algorithm with or without multi-objective optimization (Pareto). You can select the algorithm to use and its particular parameters of execution and also the subjective restrictions that the rules have to fulfil (see Figure 4), so that the rules finally shown to the user are really interesting for him.

[Insert Figure 4]

- Results: this window appears automatically after finishing the algorithm execution and lets us visualize all the discovered prediction rules (see Figure 5). Concretely, for each discovered prediction rule the conditions of the rule antecedent and consequent are shown and then all the values for each rules evaluation measure (Tan & Kumar, 2000) (Lavrac et al, 1999) (confidence, support, interest, gini, laplace, etc., currently 40 available measures). In a predetermined way, they appear ordered from the first discovered one to the last one, but they can be rearranged taking into account a condition or the value of any measure by simply clicking the desired column.

[Insert Figure 5]

To carry out the tests we have used the log information of 50 students of Computer Science Engineering at the Cordoba university taking the aforementioned course about the LINUX operating system, served through the (modified) AHA! system. Different tests have been carried out in order to

compare the results that each implemented algorithm produces in the task of knowledge discovery. The objective was to compare the number of discovered rules in each case and the quality of them based on some measures (accuracy, interestingness and comprehensibility). Because evolutionary algorithms are not deterministic, the evolutionary algorithms have been executed 10 times, and we have used the average values of all the executions. Furthermore, three different tests have been carried out: first using all data, then only the frequent data (those with a support higher than 0.5) and finally, the range data (those with a support higher than 0.2 and lower than 0.9).

[Insert Table 1]

[Insert Table 2]

[Insert Table 3]

The obtained results show (see Table 1, 2, 3) that, in general, evolutionary algorithms generate a lower number of rules but with higher interest than classic algorithms, making them more suitable to be used for the extraction of knowledge in an adaptative system of education. Classic algorithms, and above all Apriori, produce very exact rules, but fail when generating rules with a higher interest and, furthermore, the length of the produced rules makes these rules difficult to comprehend. In addition, when we use all data (that could happen when the user wants to extract global information about the system without applying any type of restriction over the group) it generates a group of rules so large that it becomes impossible to exploit the rules later. The obtained results using the proposed evolutionary algorithms show that, in general, the algorithms produce a lower number of rules than classic algorithms. Moreover, the use of algorithms based on Pareto Front (MOGA and NSGA) let us optimize the three set out objectives at the same time, producing in all the executions the higher proportion of exact, comprehensible and interesting rules.

4 Adding an author tool to create classic and adaptive computerized test

Tests or quizzes are among the most widely used and well-developed tools in higher education (Brusilovsky, 1999). There are different types of test, depending on the type of questions (yes/no questions, multiple-choice/single-answer questions, fill-in questions, etc.) and there are two types of control algorithms: adaptive and classic. A classic test is a sequence of simple questions. Each question can be evaluated as correct, incorrect or incomplete. Normally the same questions are shown to all examinees and the final score depends on the number of correct answers. An adaptive test (Arroyo, 2001) is a computer-based test where the decision of presenting a question or item and the decision to finish the test are dynamically made depending on the examinee's performance in previous answers. The main advantage of adaptive tests is that each examinee usually receives different questions and their number is usually smaller than the number of questions needed in a classic test.

During the development of the Linux course it became apparent that the creation and maintenance of the tests (to determine the entry level and the final knowledge of each chapter) was difficult and laborious. In order to simplify test creation and maintenance, we have developed an authoring tool for building both adaptive and classic web-based tests. We have integrated it in the AHA! system (De Bra, 2003). The Test Editor tool (Romero et al. 2004) does some preparatory steps: item creation, test creation and item and test maintenance. The items file creation is the first step in which examiner creates items and their parameters. Because each item is associated with one concept, the author has to select which AHA! concept is associated with this items file. Then, he adds questions to the file, one by one (see Figure 6), setting some obligatory parameters (the enunciate flag, answers and a flag to indicate whether the answers are correct) and other optional parameters (image, explanation and Item Response Theory (IRT) parameters (Wainers, 2000): item difficulty, discrimination and guessing). Items are stored in XML files. More items can be added later, and the existing ones can be modified or deleted.

[Insert Figure 6]

During the test file creation the examiner can configure tests by adding items and setting test parameters. First he has to decide what test type (classic test or adaptive test) he wants and whether to use one or several items files. If the test evaluates only one concept, we can also consider it to be an "activity". If the test evaluates several concepts, it will be an "exam" about one chapter or 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 parameters (see Figure 7) about how questions are shown to examinees (the order in questions and answers are shown, to show or hide explanations of the answers (through the "verbose" flag), the maximum time to respond, whether show the correct answer or just a score, etc.) and parameters about evaluation (to penalize incorrect answers, to penalize questions without an answer and what percentage of knowledge the final score represents in the associated concept/concepts).

[Insert Figure 7]

Finally if the test is adaptive, he has to set also the adaptive algorithm parameters (questions selection procedure and termination criterion). Each test is stored in an XML file that is prepared to be used by tests execution engine. In Figure 8, we show an example of test execution inside the AHA! tutorial. After a large number of examinees performed tests, examiners can do maintenance by carrying out modification in items and tests files. In order to facilitate this task, the Test Editor shows statistical information about examinees' usage (questions success percentage, mean times to answer the questions, questions usage percentage, etc.) to help examiners in making decisions. The examiner usually decides to modify or delete bad items, and to add new items. But he can also modify the test configuration and he can even allow certain examinees to repeat certain tests, or to allow or deny access to certain tests.

[Insert Figure 8]

4 Adding SCORM content to AHA!

Interoperability among e-learning contents and system components is a key to the successful implementation of an e-learning environment. Developing e-learning would be easier and less costly if we could just assemble courses from off-the-shelf components (Horton, 2000). The essence is that educational resources could be broken into modular components for later combination by instructors. Today there are several centres for educational technology (CETIS, http://www.cetis.ac.uk), organizations and standards (Fallon, 2002) such as: AICC (Aviation Industry CBT Committee, http://www.aicc.org) which have published standards for course management systems and the modules they control; IMS (IMS Global Learning Consortium, http://www.imsglobal.org) which is developing a more sophisticated set of standards; IEEE LTSC (Learning Technology Standards Committee, http://ltsc.ieee.org) which specifies a standard set of learning object metadata elements and SCORM (Shareable Content Object Reference Model, http://www.adlnet.org) the Department of Defence ADL (Advanced Distributed Learning) Initiative. SCORM (Faulkner, 2003) describes the components used in a learning experience, how to package those components for exchanging them between systems, how to describe them to enable search and discovery and how to define sequencing rules for the components (SCORM Content Aggregation Model). SCORM combines the AICC API specification for browser-parent frame communication with the IMS specification for content packaging and it is the most accepted specification to share contents, so that its adoption seems to be crucial in next years.

SCORM contents are packaged into PIF Files, more precisely, files according to the RFC1951, or commonly known as Zip files (Faulkner, 2003). PIF files always contain a descriptor file (imsmanifiest.xml) where content organization and resources are described and referenced. On the other hand, AHA! contents (De Bra, 2003) are organized into a root directory (only one in each course), but such directory only contains the course resources. Usually, this directory already contains course configuration files, introduction and registration files, but the information about organization and relationship between different concepts are managed by the own AHA! system itself. As we can see, content organization seems to be the key factor to allow importing and exporting facilities, since content

is presented in similar way in both systems. So, in order to import contents from a SCORM source, we need to extract SCORM course organization and inform the AHA! system about this organization. From the 'imsmanifiest.xml' file of a SCORM course we can obtain the different 'content items', which are linked to a content resource. In AHA!, content organization is managed by AHA! itself. An AHA! course (De Bra, 2003) has a 'Concept List', linking each concept to a resource (generally, an HTML file). Moreover, AHA! concepts may be organized as a hierarchy. (In the AHA! structure a concept has a relationship with its 'parent', 'next sibling' and 'first child' concept.) So, in order to import content into AHA!, we just need to extract the Zip file into a new directory, and create an AHA! 'Concept List' using the SCORM manifest file as the source. We have developed an Upload SCORM tool for AHA! (Romero et. al, 2005), that just needs a SCORM Zip file and the course name (see Figure 9), which will already be the AHA! Web-tree directory where course content is going to be extracted.

[Insert Figure 9]

When the content of a course is extracted, a directory with the same name of the course is created into the AHA! directory and the organization of the course (a 'Concept List' structure) is sent to AHA!. Then we can see the SCORM course within AHA! (see Figure 10). AHA! and SCORM environments have different characteristics and so, sometimes, once we have imported a course, it may be that its associated behaviour (evaluation and/or navigation rules) does not correspond exactly with the original author needs. So, authors should always check the imported course behaviour and they change it if necessary.

[Insert Figure 10]

To export content, we need to choose a specific AHA! course and construct an 'imsmanifiest.xml' file using the content organization information managed by AHA! and all of the AHA! resources associated with the course, forming an SCORM Zip file (Faulkner, 2003). Currently we are working in this tool for exporting AHA! courses to SCORM courses. Export should not be limited to a simple conversion of files. SCORM sequencing rules should be treated in a suitable way. So, if the way of user navigation is determined by the SCORM course, AHA! should respect such way. And, on the other hand, if navigation is not limited in SCORM course, AHA! should provide tools to allow the final user of the web-based course a free navigation. Of course, a 'recommended' next concept should be provided, to allow final user not to feel lost in the hierarchy of concepts, as it is shown (Figure 4). In addition, in order for a SCORM course to be fully integrated with AHA! educational and collaborative tools should also become available, such as a chat, an announcement board, or a download/upload system where the end-user may obtain additional content related to the course.

5 Conclusion

In this paper we have described some extensions of the AHA! system developed in the Cordoba University. 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 headed by prof. dr. Paul De Bra. At the present time the current version of AHA! is 3.0. We have modified AHA! 1.0 and AHA! 3.0 in order to increase its adaptation power in e-learning. We have modified AHA! version 1.0 to create an adaptive course in which the course material can be presented in three levels of difficulty. We have also added to AHA! version 1.0 a specific data mining tool, named EPRules, that lets to make easier the execution of rule discovery process of students' usage data. We have added Test Editor, an authoring tool to facilitate the development and maintenance of different types of multiple-choice adaptive and classic web-based test. And finally we have added an uploading tool to import SCORM contents into AHA!.

Currently we are working on exporting AHA! courses to SCORM courses, and we are developing several typical collaborative services like a chat, an announcement board, an upload tool, etc. that can be configured to be shown in the AHA! course interface. In the future, we want to provide to the AHA! Authors with a new high level tool, named 'Course Editor', where all current AHA! Author tools

(Concept Editor, Graph Editor, Test Editor, etc.), including import and export tool, can be integrated. In this way using this tool, authors would easily to create and maintain AHA! full courses.

Acknowledgement

The authors gratefully acknowledge the financial support provided by the Spanish Department of Research of the Ministry of Science and Technology under TIC2002-04036-C05-02 Projects.

Bibliography

AGRAWAL, R., IMIELINSKI, T., SWAMI, A., 1993. Mining association rules between sets of items in large databases, *ACM SIGMOD International Conference on Management of Data*.

ARROYO, I. CONEJO, R. GUZMAN, E. WOLF, B.P., 2001. An Adaptive Web-based Component for Cognitive Ability Estimation. *Proceedings of Artificial Intelligence in Education.* Amsterdam:IOS. 456-466.

BRUSILOVSKY, P. MILLER, P., 1999. Web-based Testing for Distance Education. *Proceeding of World Conference of WWW and Internet*. Honolulu:HI. 149-154.

BRUSILOVSKY, P., 2003. Adaptive Educational Hypermedia. *Proceeding of Tenth International PEG Conference*. 8-12.

CENDROWSKA, J., 1987. PRISM: an algorithm for inducing modular rules. *Journal of Man-Machine Studies,* 27, 349-370.

DE BRA, P. AND CALVI, L., 1998. AHA! An open Adaptive Hypermedia Architecture. *The New Review of Hypermedia and Multimedia*, 4, 115-139.

DE BRA, ET AL., 2003. !AHA The Adaptive Hypermedia Architecture. Proceedings of the ACM Hypertext Conference. Notttingham, UK. 81-84.

FALLON, C. & BROWN, S., 2002. E-Learning Standards: A Guide to Purchasing, Developing and Deploying Standards-Conformant E-Learning. St. Lucie Press.

HORTON, W., 2000. *Designing Web-Based Training*. United States of America: John Wiley & Sons.

FAULKNER, 2003. *Shareable Content Object Reference Model (SCORM).* Faulkner Information Services. E-doc Technology Report. 1 March.

LAVRAC, N., FLACH, P., ZUPAN, B., 1999. Rule Evaluation Measures: A Unifying View, *ILP-99*, Berlin Heidelberg.

MURRAY, T. ET AL, 2003 Authoring Tools for Advanced Technology Learning Environments: Toward Cost-Adaptive, Interactive and Intelligent Educational Software. Kluwer Academic Publishers.

QUILAN, J.R., 1987. Generating Production rules from decision trees, *Proceeding of IJCAI-87.*

ROMERO, C., DE BRA, P., VENTURA S., & DE CASTRO, C., 2002. Using Knowledge Levels with AHA! For Discovering Interesting Relationship. *World Conference on E-Learning in Corporate, Government, Healthcare, & Higher Education*. Montreal. 2721-2722.

ROMERO, C., CASTRO, C., VENTURA, S. & DE BRA, P., 2003. Discovering Prediction Rules in AHA Courses!. *User Modeling*. Johnstown, PA, USA. 25-34.

ROMERO, C., DE BRA, P., PALOMO, S., VENTURA, S., 2004. An authoring tool for web-based adaptive and classic tests. *World Conference on E-Learning in Corporate, Goverment, Healthcare & Higher Education*. Washington. 174-177.

ROMERO, C., RÍDER, J.J., VENTURA, S., HERVÁS, 2005. AHA! meets SCORM. *IADIS ELearn*. Virtual Multiconference on Computer Science and Information Systems.

SPILIOPOULOU, M., 2000. Web Usage Mining for Web Site Evaluation, *Communicacion of the ACM*.

TAN, P., KUMAR, V., 2000. Interesting Measures for Association Patterns. *Technical Report TR00-036,* Department of Computer Science, University of Minnesota.

WAINER, H., 2000. *Computerized Adaptive Testing: A premier*. New Jersey: Lawrence Erlbaum Associates.

WITTEN, I.H., FRANK, E., 1999. Data Mining. *Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann.

ZAÏENE, O.R., 2001. Web Usage Mining for a Better Web-Based Learning Environment, *Technical Report*.

ZYTKOW, J., KLOSGEN, W., 2001. Handbook of Data Mining and Knowledge Discovery, *Oxford University Press*.

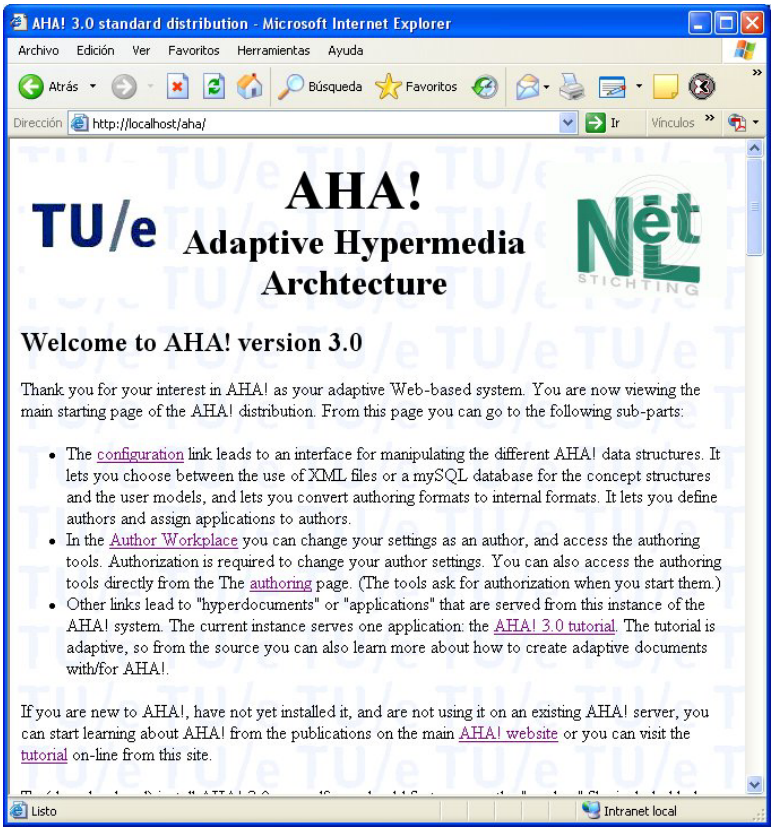


Figure 1: AHA! welcome web page.

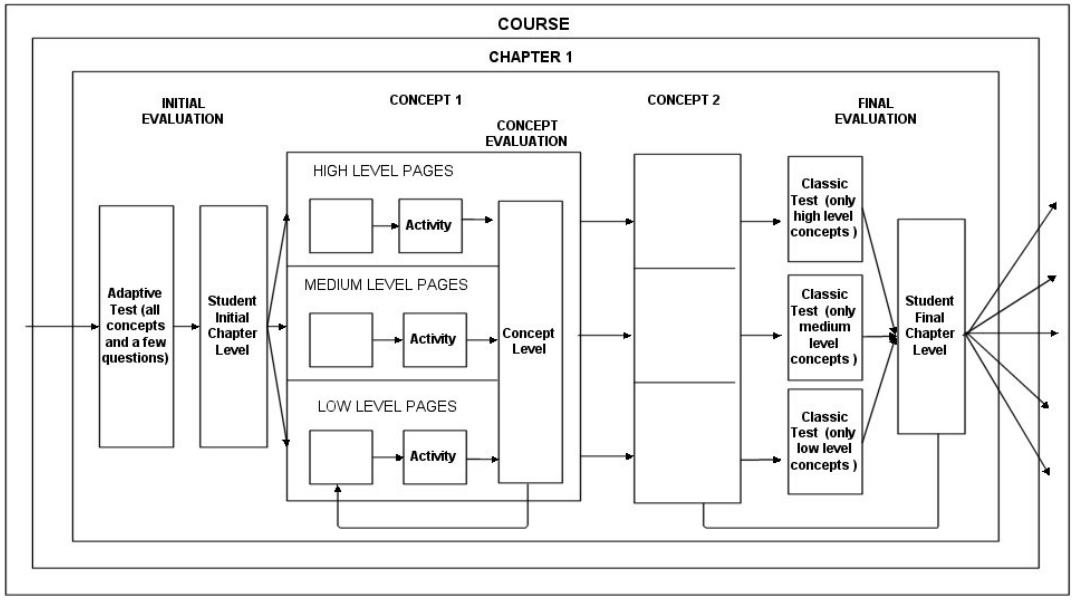


Figure 2: Modified AHA! engine.

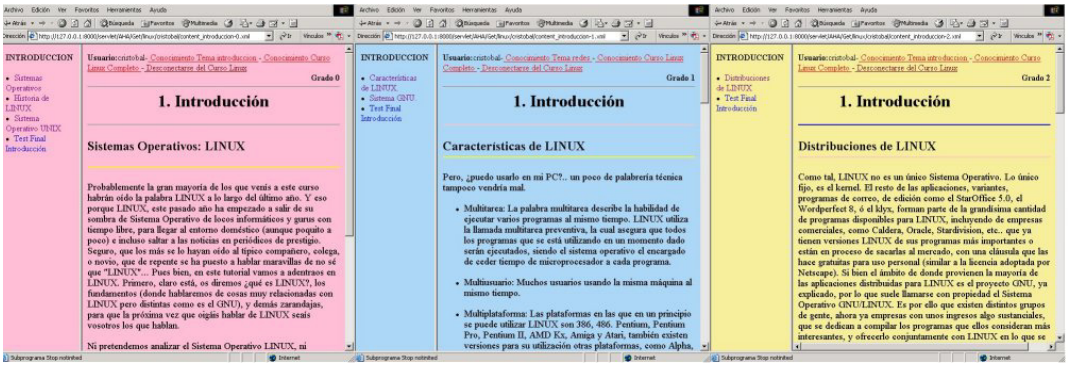


Figure 3: Three different levels of the Introduction Chapter of our Linux course.

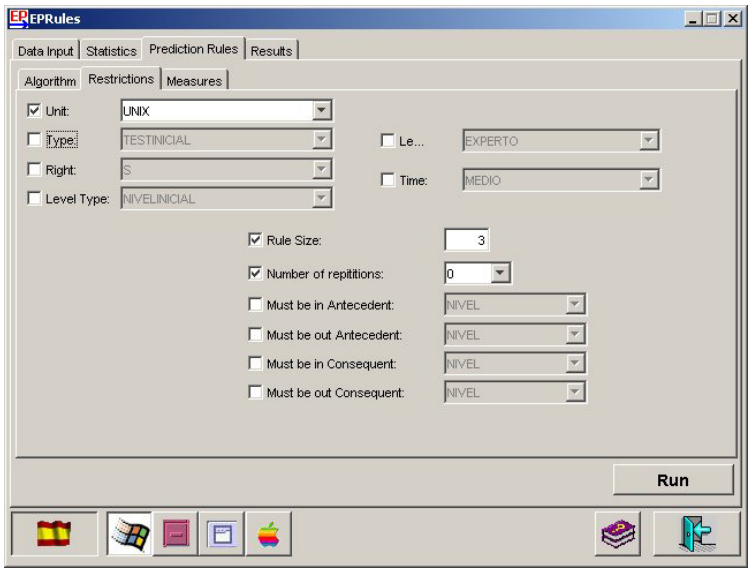
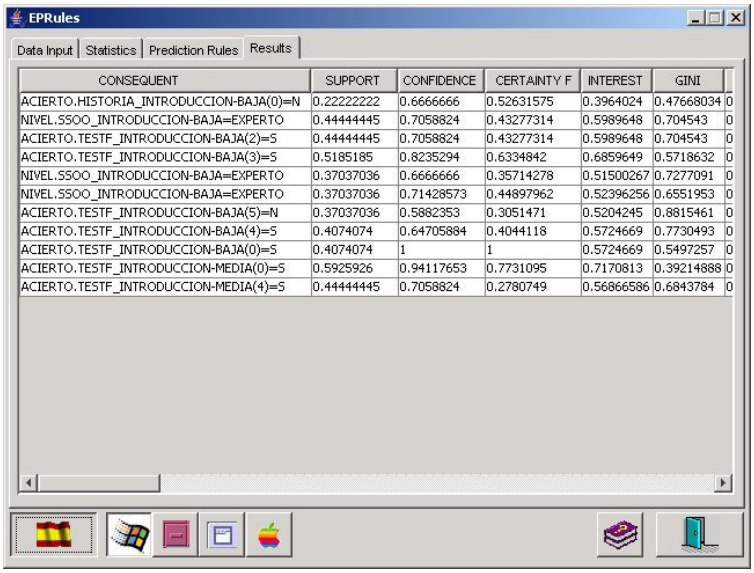


Figure 4: EPRules tool: Restrictions window.



CONSEQUENT	SUPPORT	CONFIDENCE	CERTAINTY F	INTEREST	GINI	
ACIERTO.HISTORIA_INTRODUCCION-BAJA(0)=N	0.22222222	0.6666666	0.52631575	0.3964024	0.47668034	0
NIVEL.SSOO_INTRODUCCION-BAJA=EXPERTO	0.44444445	0.7058824	0.43277314	0.5989648	0.704543	0
ACIERTO.TESTF_INTRODUCCION-BAJA(2)=S	0.44444445	0.7058824	0.43277314	0.5989648	0.704543	0
ACIERTO.TESTF_INTRODUCCION-BAJA(3)=S	0.5185185	0.8235294	0.6334842	0.6859649	0.5718632	0
NIVEL.SSOO_INTRODUCCION-BAJA=EXPERTO	0.37037036	0.6666666	0.35714278	0.51500267	0.7277091	0
NIVEL.SSOO_INTRODUCCION-BAJA=EXPERTO	0.37037036	0.71428573	0.44897962	0.52396256	0.6551953	0
ACIERTO.TESTF_INTRODUCCION-BAJA(5)=N	0.37037036	0.5882353	0.3051471	0.5204245	0.8815461	0
ACIERTO.TESTF_INTRODUCCION-BAJA(4)=S	0.4074074	0.64705884	0.4044118	0.5724669	0.7730493	0
ACIERTO.TESTF_INTRODUCCION-BAJA(0)=S	0.4074074	1	1	0.5724669	0.5497257	0
ACIERTO.TESTF_INTRODUCCION-MEDIA(0)=S	0.5925926	0.94117653	0.7731095	0.7170813	0.39214888	0
ACIERTO.TESTF_INTRODUCCION-MEDIA(4)=S	0.44444445	0.7058824	0.2780749	0.56866586	0.6843784	0

Figure 5: EPRules tool: Results window.

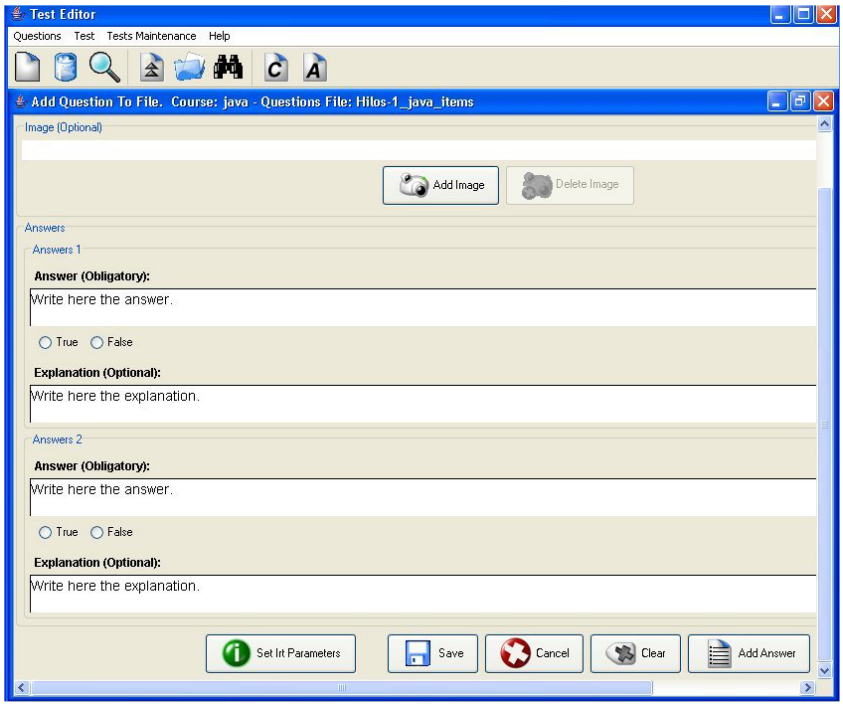


Figure 6: TestEditor: Interfaces of Items file creation.

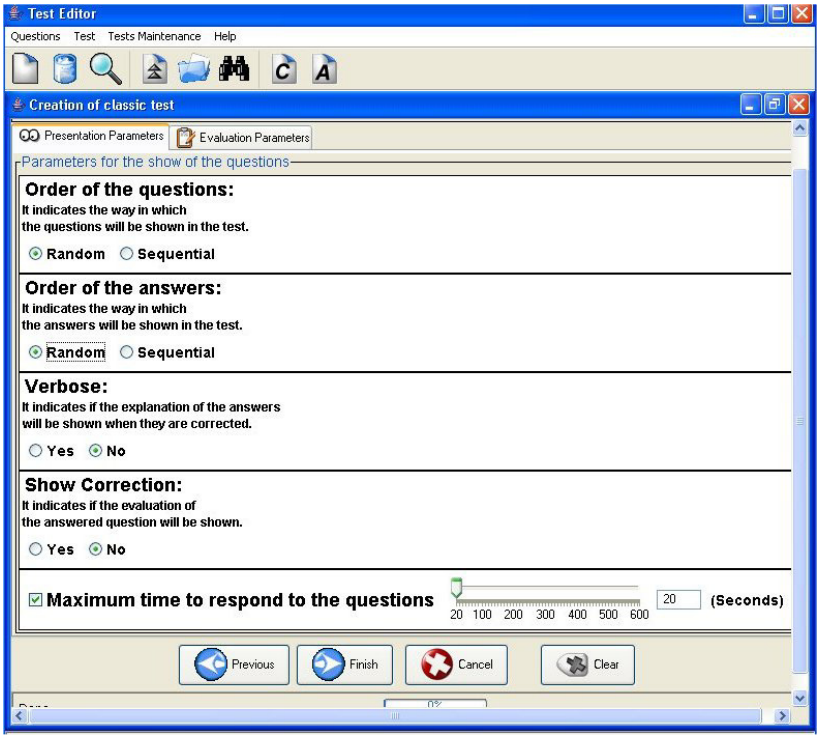


Figure 7: TestEditor: Interfaces of Test file creation.

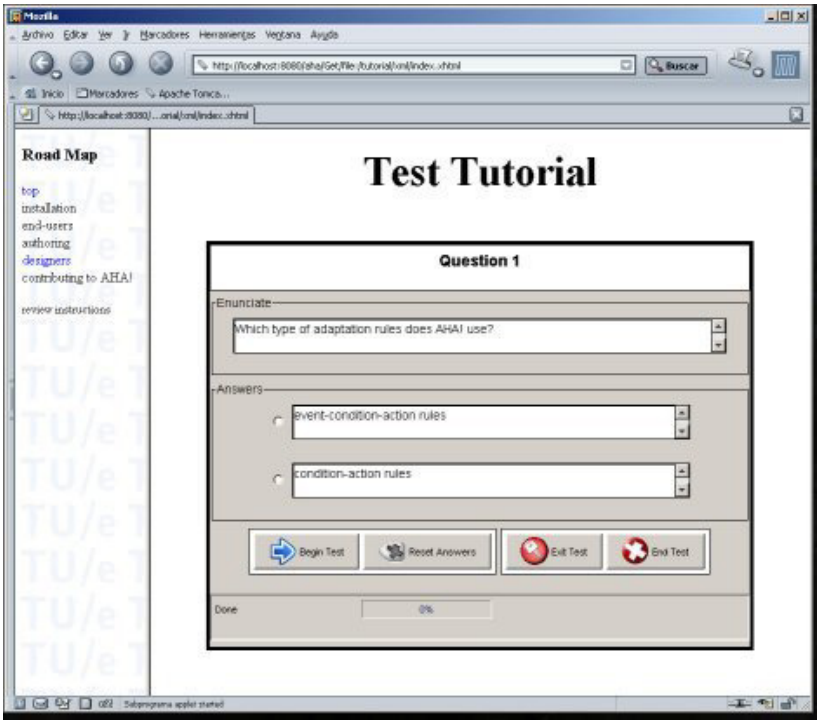


Figure 8: Example of test execution inside AHA!.

Figure 9. AHA! tool to import SCORM contents.

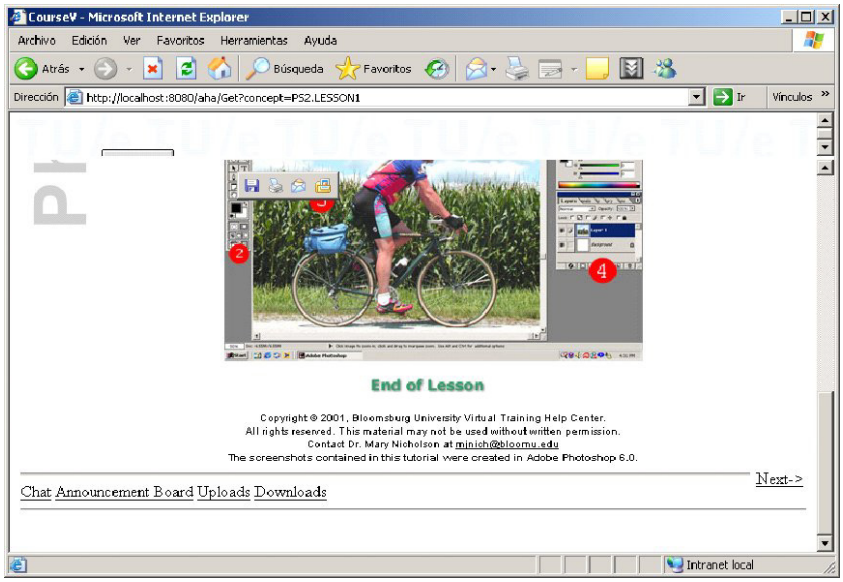


Figure 10. AHA! showing a SCORM course with limited navigation.

	Number of discovered rules		
Algorithm	**All**	**Range**	**Frequent**
ID3	474	131	89
Prism	657	172	62
Apriori	7960	491	70
AE-GBGP	198	162	51

Table 1: Comparison of the number of discovered rules.

Algorithm	Percentage of exact rules		
	All	Range	Frequent
ID3	46,0	51,9	60,3
Prism	71,9	53,7	91,9
Apriori	84,3	90,0	93,0
AE-GBGP	76,5	86,1	96,3

Table 2: Comparison of the percentage of exact rules discovered.

Algorithm	Percentage of interesting rules		
	All	Range	Frequent
ID3	1,5	7,6	15,6
Prism	2,5	11,6	49,3
Apriori	3,6	7,9	53,1
AE-GBGP	21,9	60,4	76,6

Table 3: Comparison of the percentage of interesting rules discovered.