An Authoring Tool for Web-Based Adaptive and Classic Tests

Cristóbal Romero[1], Santiago Martín-Palomo[1], Paul De Bra[2], Sebastián Ventura[1]

[1]Departament of Computer Science, University of
Cordoba. 14071 Campus de Rabanales,
Cordoba, España.
{cromero,i02pagas,sventura}@uco.es

[2]Department of Computer Science, Eindhoven
University of Technology (TU/e) PO Box 513,
Eindhoven, The Netherlands.
debra@win.tue.nl

Abstract: In this paper we describe Test Editor, an authoring tool for building adaptive and classic web-based test. This tool facilitates the development and maintenance of different types of multiple-choice tests for use in web-based education systems. Test Editor is a modular tool which lets you configure several parameters about items and tests. It also provides statistical information about tests usage that can be used in test maintenance. We have integrated the Test Editor with the AHA! system, but it can be used in other web-based systems as well.

Introduction

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.

In order to facilitate test creation and maintenance, we have developed an authoring tool for building both adaptive and classic web-based tests. Currently, we have integrated it in the AHA! system (De Bra, 2003) because it is a well-known adaptive hypermedia architecture used to build web-based courses, and because it uses an XML-based markup language. The specific life cycle of tests we have used is shown in Figure 1.

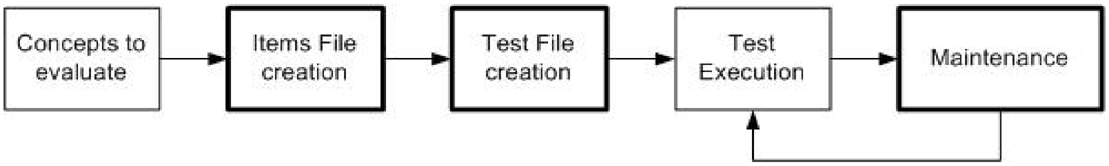
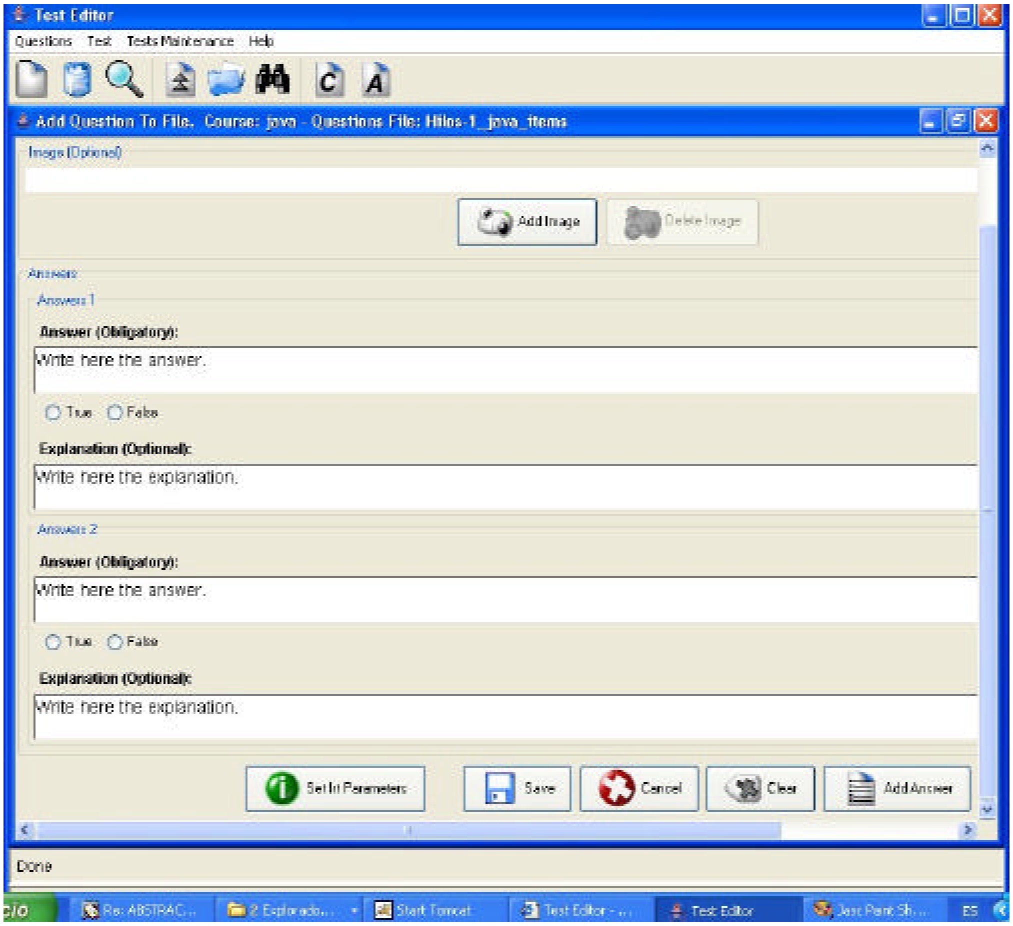


Figure 1: Life cycle of tests.

The first step is to specify what the concepts are we want to evaluate. In AHA! for example we can use Graph Editor or the Concept Editor (DeBra, 2003) to create these concepts. The second step is to build items of the test. Each item is one question about one concept and its parameters (question, answers, explanation, etc.). The third step is to build tests out of items. Tests consist of a number of items and configuration parameters (the type of test, number of questions, etc.). Then tests are published andthen performed by examinees usinga test execution engine. In a final step, tests and items can be modified by the examiner based on the examinees' usage information. The Test Editor tool does some preparatory steps: item creation, test creation and item and test maintenance (see Figure 1 with underlined square). Next, we are going to describe each step.

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 2), 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 and more items can be added, and the existed ones can be modified or deleted.



Answers 1

Answer (Obligatory):

Write here the answer.

○ True ○ False

Explanation (Optional):

Write here the explanation.

Answers 2

Answer (Obligatory):

Write here the answer.

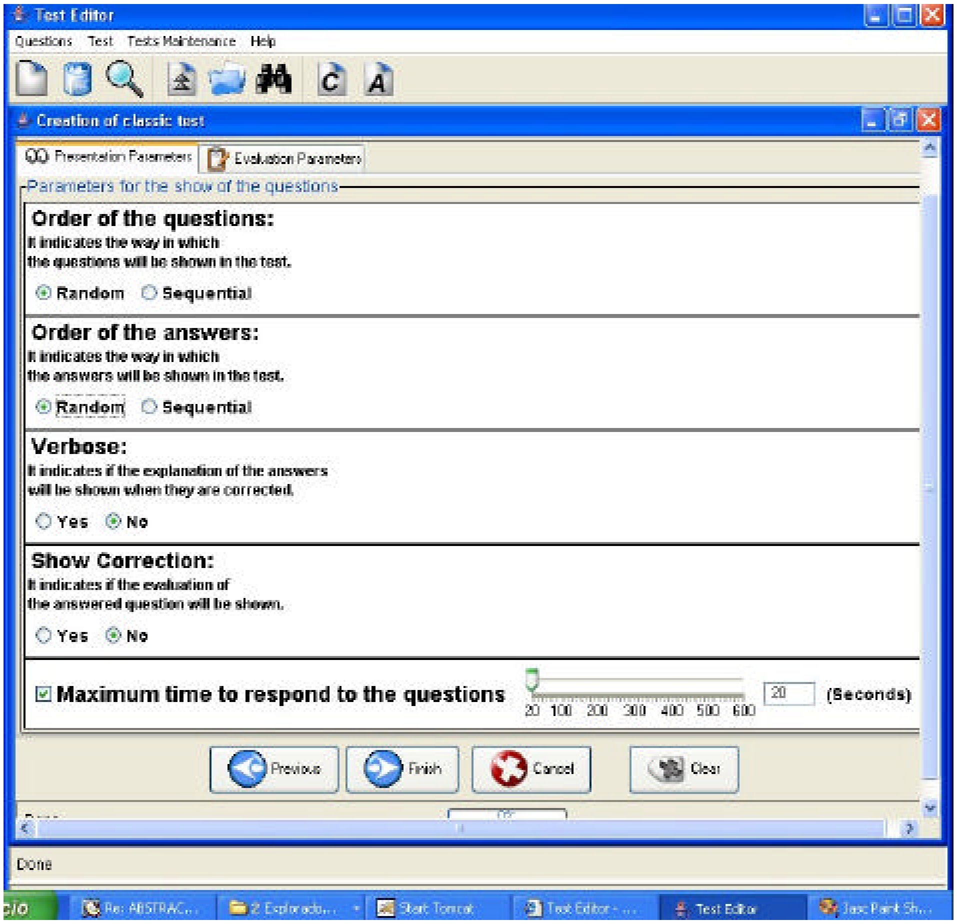
○ True ○ False

Explanation (Optional):

Write here the explanation.

Figure 2: Interfaces of Items file creation.

During the test file creationthe examiner can configure tests by adding items and setting test parameters. Firsthe 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 3) 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 withoutan answer and what percentage of knowledge the final score represents in the associated concept/concepts).



Parameters for the show of the questions

Order of the questions:
It indicates the way in which
the questions will be shown in the test.
● Random ○ Sequential

Order of the answers:
It indicates the way in which
the answers will be shown in the test.
● Random ○ Sequential

Verbose:
It indicates if the explanation of the answers
will be shown when they are corrected.
○ Yes ● No

Show Correction:
It indicates if the evaluation of
the answered question will be shown.
○ Yes ● No

☑ **Maximum time to respond to the questions** 20 100 200 300 400 500 600 | 20 | (Seconds)

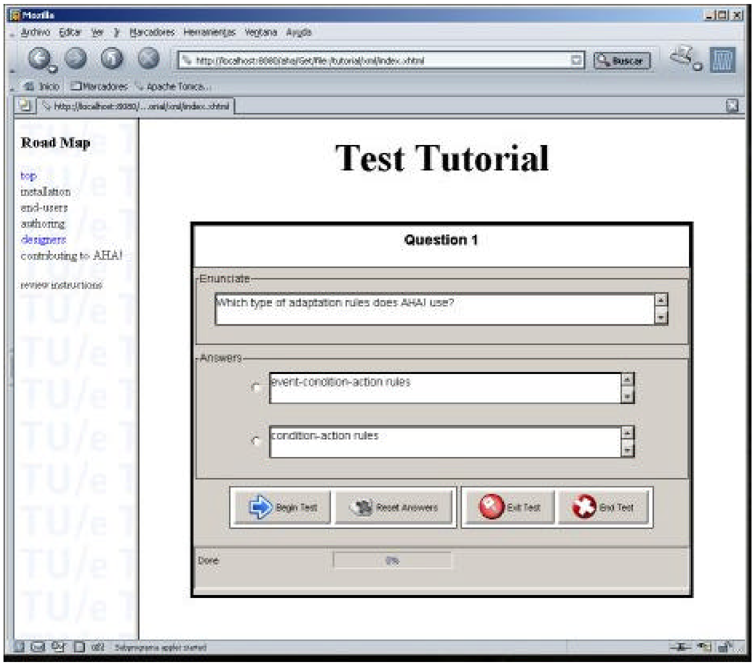
Figure 3: Interfaces of Test file creation.

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 the Figure 4, 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.

Conclusions

The Test Editor is a tool for the development and maintenance of different types of web-based test. The main advantages of Test Editor are: modular (separate concepts, items and tests), easy to use (Java Swing graphic user interface); it facilitates the maintenance (based on examinees' usage information) and uses a standard format (xml files). Although we have created it within the AHA! system, it can be used in other web-based educational systems. The only requirement is that the test execution engine can read and show the XML tests files as produced by the Test Editor.



Test Tutorial

Question 1

Enunciate

Which type of adaptation rules does AHA! use?

Answers

○ event-condition-action rules

○ condition-action rules

Figure 4: Example of test execution inside AHA!.

References

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

De Bra, P., Aerts, A., Berden, B., De Lange, B., Rousseau, B., Santic, T., Smits, D., Stash, N. (2003) The Adaptive Hypermedia Architecture. *Proceedings of the ACM Hypertext Conference* (pp.81-84). Notttingham:UK.

Brusilovsky, P. Miller, P. (1999) Web-based Testing for Distance Education. *Proceeding of World Conference of WWW and Internet* (pp. 149-154). Honolulu:HI.

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* (pp. 456-466). Amsterdam:IOS.