An Authoring Tool for Building Both Mobile Adaptable Tests and Web-Based Adaptive or Classic Tests

Cristóbal Romero[1], Sebastián Ventura[1], Cesar Hervás[1], and Paul De Bra[2]

[1] Córdoba University, Campus Universitario de Rabanales, 14071, Córdoba, Spain
{cromero, sventura, chervas}@uco.es
[2] Eindhoven University of Technology (TU/e), PO Box 513, Eindhoven, The Netherlands
debra@win.tue.nl

Abstract. This paper describes Test Editor, an authoring tool for building both mobile adaptable tests and web-based adaptive or classic tests. This tool facilitates the development and maintenance of different types of XML-based multiple-choice tests for using in web-based education systems and wireless devices. We have integrated Test Editor into the AHA! system, but it can be used in other web-based systems as well. We have also created several test execution engines in Java language in order to be executed in different devices such as PC and mobile phones. In order to test them, we have carried out two experiments with students to determine the usefulness of adaptive tests and mobile tests.

1 Introduction

Computerized tests or quizzes are among the most widely used and well-developed tools in web-based education [7]. 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 [20]. The main advantage of computerized adaptive tests (CAT) is that each examinee usually receives different questions and their number is usually smaller than the number of questions needed in a classic test. Currently, there are several well-known commercial and free tools for developing adaptive and classic computerized test such as: QuestionMark [14], Webassesor [19], MicroCAT and FastTEST [2], SIETTE [1], Test++ [5], etc. Most of them are based on XML to record the information about assessments and some use the IMS Question and Test Interoperatiblity (QTI) international specification [4]. On the other hand, m-learning (mobile learning) and u-learning (ubiquitous learning) have started to emerge as potential educational environments [11]. In fact, there are nowadays several quiz systems [10] oriented to be used not only for PC users, but also for PDA and mobile phone users; and there are some interactive tests [12] specifically developed only for being used in mobiles phones. There are also several commercial tools such as Mobile EMT-B quiz [13], oriented to PDA devices and others such as Go Test Go's [9] oriented to be used in Java mobile phones. With the Test Editor described in this paper it is possible to author once and deliver on both mobile and Web-based platforms.

V. Wade, H. Ashman, and B. Smyth (Eds.): AH 2006, LNCS 4018, pp. 203 – 212, 2006.
© Springer-Verlag Berlin Heidelberg 2006

2 Test Editor Author Tool

In order to facilitate computer tests creation and maintenance, we have developed the Test Editor tool for building computerized tests [16]. Currently, we have integrated it in the AHA! system [8] because that is a well-known adaptive hypermedia architecture used to build web-based courses, and because it uses the Java and XML languages. Test Editor is a (signed) Java Applet, just like other AHA! authoring tools: Form Editor, Concept Editor and Graph Editor.

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 and wrong) and explanations for the wrong answers. Several items/questions about the same concept can be grouped together into one items file. Figure 1 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 [20]: 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 (currently only AHA! 1.0 and AHA! 3.0). Questions can thus be re-used from other test environments without needing to enter them again.

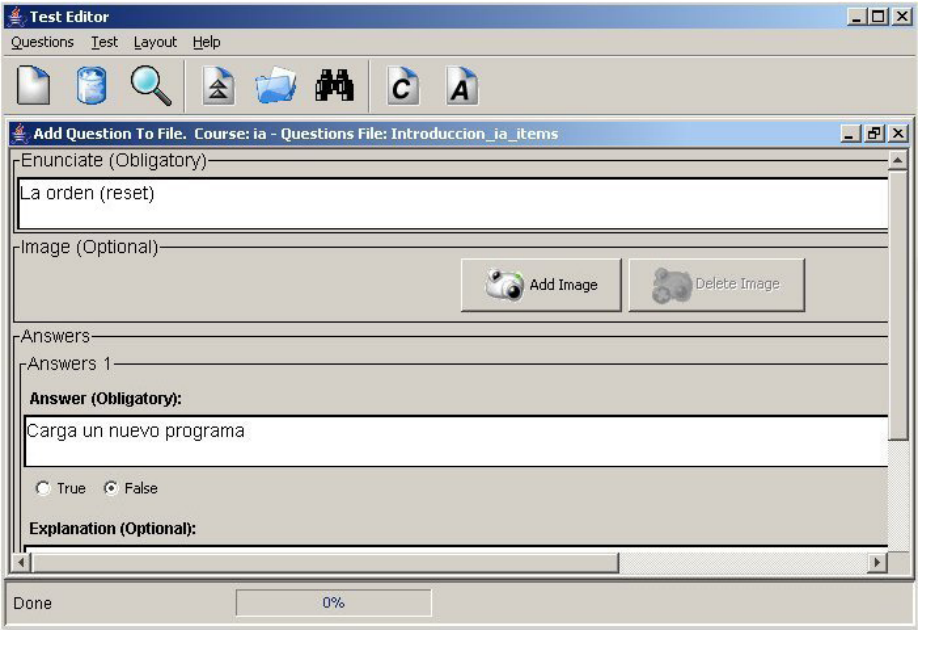

Fig. 1. Test Editor: Windows to introduce the obligatory parameters of an item

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 2) 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 [21] that includes both the multiple-choice test code (a Java Midlet test engine) as well as the questions and parameters (XML file).

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 [8]. When used with AHA! a test is presented in an Java Applet, with a look and feel that is similar to the Java

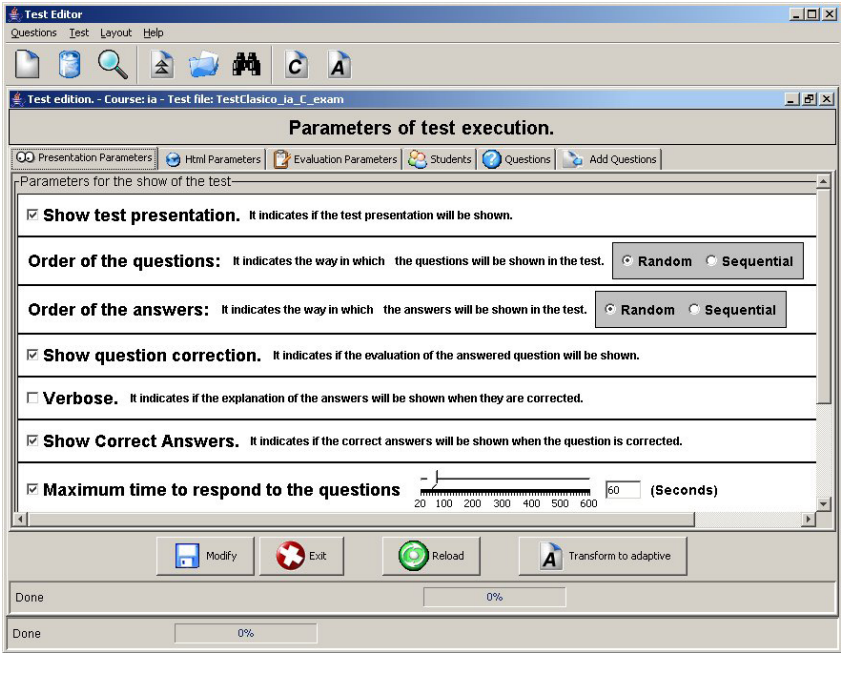

Fig. 2. Test Editor: Windows to select the questions presentation parameters

Midlet version. 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/improvements to 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 [3], in order to transform a classic test into an adaptive one, or to optimize the IRT parameter of an adaptive test.

3 The Web-Based Adaptive and Classic Tests Engine

Our web-based tests engine is a signed Java Applet that uses Java Servlets to communicate with AHA! [8]. It can execute both classic and adaptive computerized tests with multiple-choice items [16]. A conventional (classic) test is a sequence of simple questions and normally the same questions are shown to all examinees. The algorithm to control the execution of a classic test is very simple: it shows a sequence of questions until either there are no more questions or the examinee has used up the maximum allowed time. On the other hand, a CAT [18] is a computer-based test where the decision about presenting a question or item and finishing the test is made depending on the examinee's performance in previous answers. The general adaptive tests algorithm (see Figure 3) consists of three main procedures: question selection, based on the most informative item for each student; proficiency estimation of each student; and checking the finalization criteria (maximum number of questions, maximum spent time or if the proficiency level has passed a confidence value).

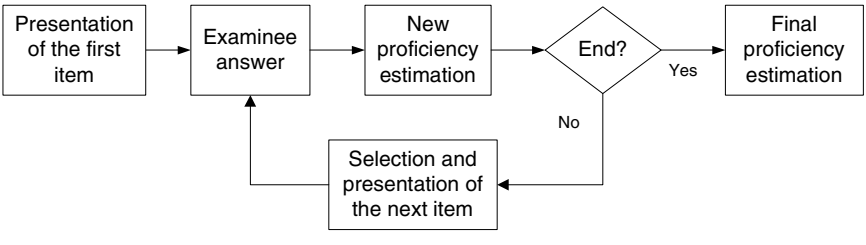

Fig. 3. Adaptive tests control algorithm

When a student starts a test (clicking on the test link), the engine connects to the server in order to obtain all the test information and to check if the student is allowed to take the test (or repeat it). If the test has "starting information" the engine will show it, and it will then start to show questions. The student has to select what the hopefully correct answer is (possibly more than one) and then presses the "Correct The Question" button (see Figure 4). This has to happen before the maximum response time has elapsed. The student can see if the submitted answer was correct or incorrect, if the author has set the parameter to show this. Finally, after the student replies to the last question he will see the obtained score and the total time spent.

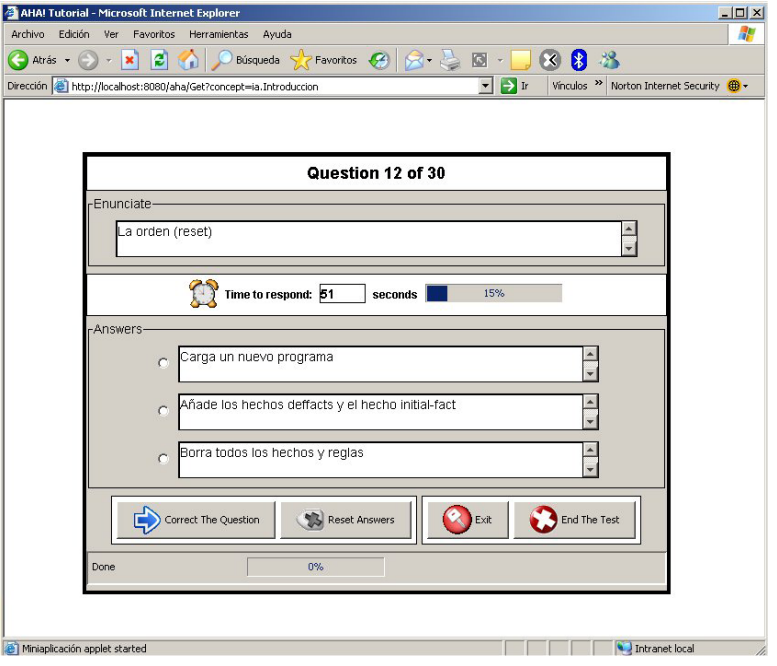

Fig. 4. Web-based tests execution engine interface with a question

4 Mobile Adaptable Tests Engine

Our mobile adaptable tests engine is a Java Midlet [21] with a specific tests interface designed for small wireless devices. Java Midlets are small applications that can be executed in the mobile phone. They have important advantages compared to WAP (Wireless Application Protocol) and browser based applications. For example, they can be used offline without connection cost, they have a more responsive and interactive interface and they are popular thanks to Java games [12]. Functionally, our mobile engine can read (XML) test files, present questions, check answers and send the score back to AHA! [8]. The user can download and install the *.jar* file (generated by the Test Editor) in the mobile device directly from Internet (by connecting to the *.jad* file), or he can download the *.jar* file to a PC first and then send it to the mobile using Bluetooth, Infrared, serial bus, etc. After installing, the execution of the test is totally off-line and it works as shown in Figure 5: the questions are shown on the mobile's screen in a linear or random order (depending on the test parameters), the answers have to be selected by the user with the phone keys and when the test ends the scores obtained and the used time are shown.

Mobile tests engine has some personalization characteristics for individualised execution [6]:

– When the user starts the application he/she has to identify himself/herself by introducing his/her personal login and password (the same as used in AHA!).

- When the user finishes the test execution the scores are physically stored in the mobile memory card by using RMS (Record Management System).
- If the user executes an exam, then the elapsed time in each question is shown.
- The user can send the obtained score to AHA! (in order to update his/her AHA! profile) through a GPRS (General Packet Radio Service) connection.
- Activities can be repeated several times by the same users, but exams cannot. The user cannot easily hack the downloaded *.jar* exam (for example, he can try to do it by uninstalling and installing the application again) because when an exam starts the application connects to AHA! in order to check that the user has never taken that exam before.

Fig. 5. Mobile tests execution interface with a question and the final score

Mobile tests engine also has some adaptable characteristics in the interface. The difference between adaptive and adaptable refers to the extent to which users can exert influence on the individualization process of a system [17]. Adaptable systems are customized by the users themselves. In our mobile tests application, the user can select the following preferences from the main menu (see Figure 6 at the left):

- The user can choose between different font types (see Figure 6 in the middle) and sizes, in order to improve the readability of the text of the questions.
- The user can choose to show questions and answers together on the same screen (see Figure 6 at the right) if he/she prefers to scroll, or to show them on two different screens (see Figure 5 at the left and in the middle) if he/she prefers to see the question on one screen and the answers on another.
- The user can choose to show the associated images that some questions have, if he/she has a screen big enough to show them, or not to show them if he/she has a small screen.

Fig. 6. Mobile tests main menu interface and preferences

5 Experimental Results

We have carried out two experiments to determine the usability of both adaptive tests (a calibrated version versus a non-calibrated version) and mobile tests (a PC version versus a Mobile version), using two different tests about the Java and CLIPS languages respectively.

In the Java Language test we compare the results students obtain when they use the same PC test but with adaptively calibrated items and with non-calibrated items. Each test has been carried out by a different group of 60 computer science engineering students at the Cordoba University, with a similar age, knowledge and experience. Both tests consisted of the same 27 items with 4 possible answers on which one answer was correct, and the same finalization conditions (if the standard error became lower than 0.33 or if all 27 questions were presented). The difference is that initially the IRT two-parameters (difficulty and discrimination) of the non-calibrated items are set manually by experts in Java, and after one group of students executes the test then the IRT two-parameters are calibrated using the maximum likelihood estimation estimator [3] to be used with the other group of students.

Table 1. Students tests execution results: adaptive non-calibrated versus calibrated test

	Time taken	Number of Items	Proficiency estimation	Standard error
Non-Calibrated Test	434.6±88.8	26.9±1.6	-1.3±0.3	0.6±0.1
Calibrated Test	182.4±81.2	11.5±2.6	-2.2±0.3	0.4±0.1

In the Table 1, we show the mean value and the confidence interval (95%) of the time taken (in seconds) to complete the test, the number of items attempted, the proficiency estimated and the standard error. We can see in the first table row, there was a reduction in the total number of questions used in the calibrated version versus the non-calibrated version. Secondly, we can see a reduction of the time needed to complete the test in the calibrated version precisely due to the reduction of questions. Finally, the estimated proficiency obtained in the calibrated version is lower than the non-calibrated version but the standard error is higher. It shows that the precision obtained in the calibrated version is higher, and the student's estimated proficiency is more accurate, as was expected.

On other hand, in the CLIPS Language test we compare the results students obtain when they execute the same test but on the PC or by the mobile phone. Each test has been carried out by a different group of 80 and 20 computer science engineering students (with Java mobile) at the Cordoba University, all with similar age, knowledge and experience. Both tests consist of the same 40 items with 3 possible answers of which one was correct. The questions were shown in random order.

Table 2. Students tests execution results: web-based classic test versus mobile test

	Time taken	Number of correct items	Number of incorrect items	Number of items without answer
PC Test	1157.8±75.2	19.8±0.8	6.3±0.6	3.8±0.5
Mobile Test	635.1±58.7	20±1.5	5.4±1.2	4.8±1.1

In the Table 2, we show the mean value and the confidence interval (95%) of the time taken (in seconds) to complete test, the number of correct items, number of incorrect items and the number of items without answer. We can see that the execution of the mobile test is much quicker than the PC test: students with the mobile test used only about half of the time that students with a PC needed. This can be because the user interface and input methods of this technology are simple and efficient (some examples are Java games and SMS applications) and so, the students show a great proficiency in using them (fast browsing through mobile interfaces). And the final scores were very similar in both versions with only small differences.

Finally, we have also carried out a survey among all the students of the CLIPS test in order to learn what their opinions are about the two versions of the test. The questionnaire had five questions (1.How much do you prefer it?, 2.How useful is it?, 3.How easy to use is it?, 4.How much do you like the user interface? and 5.How much do you like the data entry method?) that students have to answer with a range between 1 (a little) and 5 (much) for each version, and they can also write some comments.

Table 3. Student's opinion questionnaire: web-based classic versus mobile test

	More preferable	More useful	More easy to use	Best user Interface	Best data entry method
PC Test	3.57±0.34	3.78±0.55	4.78±0.18	4.05±0.23	4.36±0.37
Mobile Test	3.89±0.39	4.26±0.36	4.47±0.31	3.68±0.33	4.01±0.39

In the Table 3, we show the mean value and the confidence interval (95%) of the rating for preference, usefulness and ease of use of the test, and the rate of acceptance of the user's interface and the data entry method. We can see that the mobile test is more preferable and useful than the PC test, although the PC test is easier to use and it has a better user interface and data entry method. This can be because students are still more familiar with PC interfaces and their data entry methods for this type of applications. But, in general, students liked the experience to use a mobile application to execute tests that can evaluate their knowledge in a specific area. About the comments, students think that the main weaknesses of mobile phones are:

- Small screen size. In general, all students would prefer to be able to see questions, question and answer on the same screen and without needing to scroll although they are long, as they are written with the size of a PC screen in mind.
- Very expensive. Almost all the students think that Java mobile phones are very expensive at the moment, and it is necessary that they become cheaper in order for most of the students to be able to afford them. Once affordable the mobile tests and other m-learning tools will become really useful and usable in real life.
- Difficult input method. Some students with big fingers had some problems to press the correct button each time and they would like that mobile could have bigger buttons or some other alternative input method.

6 Conclusions and Future Work

In this paper we have described Test Editor, an authoring tool for building computerized tests. The main advantages of Test Editor in relation to other test tools are: modular (concepts, items and tests are clearly separated), easy to use (it has a friendly Java Swing graphical user interface); it facilitates the maintenance (it has statistical information and item calibration based on examinees' usage information), standard format (it uses XML files) and multi-device execution (it has several Java engines for executing tests on a PC and on wireless devices). We have resolved the problem of authoring once for delivery on two very different platforms using XML for storing test information and Java for developing the different test execution engines. Although we have integrated it within the AHA! system [8], it can be also used in other web-based educational systems that support the Java and XML languages. After the experimentation, the first impression is that students are generally highly motivated to use mobile technologies for testing and it can be possible and useful to use mobile devices for testing despite some limited possibilities of J2ME (Java 2 Micro Edition) such as small screen size, limited application size and no support for floating point numbers. But we have developed a user interface with preferences; we have tried to reduce the number of lines of code and we have used a Java floating point emulation library for J2ME [15].

Currently we are working on extending the interoperability with other tests formats. We want to allow import/export questions and tests to/from others computerized tests systems and standards such as IMS QTI [4], QuestionMark [14], SIETTE [1], etc. In the future, we want to add more adaptable characteristics and to develop an adaptive tests control algorithm for the test mobile engine.

Acknowledgments

The authors[1] gratefully acknowledge the financial support provided by the Spanish department of Research under TIN2005-08386-C05-02 Project.

References

1. 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
2. Assessment Systems Corporation: Microcat and Fasttest, http://www.assessment.com (2006)
3. Backer, F.: Item Response Theory, Parameter Estimation Techniques. Marcel Dekker (1992)
4. Bacon, D.: IMS Question and Test Interoperability. MSOR Connections, 3:3 (2003) 44-45
5. Barra, M., Palmieri, G., Napolitano, S., Scarano, V., Zitarosa, L.: Adaptive Testing by Test++. Proc. of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, Trento, Italy, (2000) 264-267
6. Bull, S., Reid, E.: Individualised revision material for use on a handheld computer. Proc. of the International Conference on MLEARN, UK (2003) 35-42
7. Brusilovsky, P., Miller P.: Web-based Testing for Distance Education. Proc. of the World Conference of WWW and Internet, Hawaii, USA, (1998) 149-154
8. De Bra, P., Aerts, A.. Berden, B., De Lange, B., Rousseau, B., Santic, T., Smits, D., Stash, N.: AHA! The Adaptive Hypermedia Architecture. Proc. of the ACM Hypertext Conference, Nottingham, UK, (2003) 81-84
9. Go test go: http://www.gotestgo.com (2006)
10. Goh, T., Kinshuk, Lin, T.: Developing an adaptive mobile learning system. Proc. of the International Conference on Computers in Education, Hong Kong, (2003) 1062-1065
11. Kinshuk: Adaptive Mobile Learning Technologies. GlobalEducator.com, (2003)
12. Mayorga, M.C., Fernández, A.: Learning Tools for Java Enabled Phones: An Application to Actuarial Studies. Proc. of the International Conference MLEARN, UK (2003) 95-98
13. Mobile EMT-B Quiz: http://www.emszone.com/mobilequiz (2006)
14. QuestionMark: http://www.questionmark.com (2006)
15. Real: http://sourceforge.net/projects/real-java (2006)
16. Romero, C., De Bra, P., Ventura, S.: An authoring tool for web-based adaptive and classic tests. Proc. of the World Conference on E-Learning in Corporate, Goverment, Healthcare and Higher Education, Washington, (2004) 174–177
17. Treiblmaier, H.: Measuring the Acceptance of Adaptive and Adaptable Systems. Proc. of the HHCCII, Hawai, USA, (2004) 1-8
18. Van der Linde, W. J., Hambleton, R. K.: Handbook of Modern Item Response Theory. Springer Verlag, Berlin (1997)
19. Webassessor: http://www.webassessor.com (2006)
20. Wainer, H.: Computerized Adaptive Testing: A premier. New Jersey, Lawrence Erlbaum Associates (2000)
21. Yuan, M.J.: Enterprise J2ME: Developing Mobile Java Applications. Prentice Hall, New Jersey (2003)