eWorkbook: a Web based tool for assessment and self-assessment

Gennaro Costagliola
Filomena Ferrucci
Vittorio Fuccella

[gencos, fferrucci, vfuccella]@unisa.it

This paper introduces a Web application for creating and making use of on-line tests to evaluate learner's competency in different subjects by means of multiple choice questions. Its use is suitable within the academic environment in a blended-learning fashion both by tutors, for having an additional assessment tool, and by learners, for performing a distant self-assessment. Main features, architectural design, technical notes and future extensions of our application, called eWorkbook, are covered in this paper.

1. Introduction

Blended Learning is regarded as the most effective way to train learners by the greatest part of e-learning researchers. It's more and more widely adopted, not only in situations in which distance requires its use, but also as an efficient way to support traditional face-to-face learning. At the University of Salerno, our research group, funded for this research by CampusONE Project, has been experimenting with some systems and platforms to support blended learning. This experimentation has revealed the need for an advanced assessment tool easily integrable with the existing systems.

According to the experimentation, the following features were needed by this kind of software:

- High re-usability of the authored content. This is a very important matter in the e-learning ambit, where content development is quite an onerous task.
- A customizable assessment strategy, to give to the tutor the chance to tailor the test to the competence and skill level of his/her class
- A rich reporting section, to permit, the learner to self-assess properly, and the tutor to evaluate learner's results and test effectiveness
- A flexible access control system to the tests, to prevent the learner from following the wrong learning path, denying him access to learning experiences that could require a prior study of other subjects or simpler matters.
- Full accessibility and usability of the Web-based interface: no technical grounds (restriction due to software compatibility or lack of usability standard compliance) should limit these features for any user

Several different assessment tools and applications to support blended learning have been analyzed, starting from the most common Web-based e-learning platforms: *WebCT 4.1 Campus Edition* [WebCT], *Blackboard 6* [BlackBoard], *Click2Learn Aspen 2.0* [Aspen], *EduSystem* [EduSystem], and *The Learning*

Manager 3.2 [LManager]. Our analysis was lead by means of direct experience, literature surveys and benchmark analysis. *EduTools* [EduTools] was particularly useful in the latter task.

The most useful way to make questions and tests re-usable is to have a repository from which questions can be picked-up when a test must be created. A simple use of a question pool is proposed in [Lister and Jerram, 2001], while a structured hierarchical organization of the repository (a tree organization with multilevel relations between the elements) can be found in [Gusev and Armenski, 2002]. Those systems use a classification based on the question's subject area. A more complete but complex system, described in [Proctiæ et al, 2001], not only does that (using a multi-tree schema), but also classifies questions in accordance with similarities among them.

None of the analyzed tools allows to customize the assessment strategy for each test. Some weaker features are available: Blackboard gives the tutor the chance to weight tests differentially and create grading rules. In WebCT instructors can mark all assessments not automatically scored online and manually edit all grades.

Many of the tools we considered have a learner's performance history section and a statistics section. Some of them allow the import of the results onto a spreadsheet. In most cases they limit the history and statistic feature to the consultation of the learner *gradebook*. An interesting feature is the opportunity to judge a question or a test analyzing the learners' responses to it. Starting from this information, a great deal of criteria can be adopted. I.e., Hicks [Hicks, 2002], describing his experience with a large class at the University of Newcastle upon Tyne, identifies a good question as the one to which the better 20% of students answers well and the worse 20% of students answers incorrectly. In [Lira et al, 1990] the degree of difficulty of a test is calculated using the maximum (max) and minimum possible (min) score and the average score (avg) of the class according to the following formula: ((avg – min) / (max – avg)) * 100.

As for a means to control access to the tests, none of the tools analyzed has a flexible system (i.e. a system that allows to define prerequisites for each test). The system described in [Lister and Jerram, 2001] permits the learner to sit an exam many times, until a minimum acceptable score is achieved. In [Gusev and Armenski, 2002] the questions are grouped into sets, and the strategy to pass a set, and consequently access to the next, is to give the correct response to *3 answers in a row* for that set.

Since none of the tools considered satisfied all of our requested features, eWorkbook was created to address the actual needs.

2. Features

2.1. Shared Repository

First of all, eWorkbook has a centralized question and test repository, accessible to all tutors. It contains all question and test items that have been published. Publication of an item in this shared repository is preceded by its authoring. eWorkbook supports this phase allowing the tutor to produce different items in a draft state. When the author is satisfied of the item's behaviour he can activate and share it with the other tutors.

The shared repository is partitioned in semantic macro-areas: each of them is a container of questions that holds items of a specific subject. The partition can be fine or coarse according to tutors' will. The tutors all have the same permission set on the repository and they can add or delete macro-areas and questions.

In order to get better results, it's a good practice to initialize the repository with a wide set of macro-areas and questions before the system starts to be operating fully.

2.2. Tests

The test authoring process is quite simple but powerful and it is aided, as all other procedures provided by eWorkbook, through a wizard that consists of a sequence of screens where the user can choose some behavioural features. In this case, the tutor has to set the basic test features (number of questions, maximum time to complete the test) in a first screen and the advanced ones (static or dynamic choice of the questions, later explained) in a latter and last screen. Other advanced features are not directly bound to the test, but to the course in which the test is presented. That's due to reasons that we further investigate later on.

There are two ways to select the questions that compose a test: through a static creation-time choice and a dynamic run-time choice, as the learner joins the test. A test created using the former choice is called "static test", while a "dynamic test", created using the latter choice, allows the author just to specify the percentile distribution of the questions among the macro-areas: the questions are randomly chosen at run-time.

For a static test, the tutor must choose all the questions it will present to the learner. A question selection procedure in a repository with a big size could be a very tedious matter. For this reason, eWorkbook provides an automatic method that randomly chooses the questions and submits them to the author's approval. This method is iterative: once a first sequence of questions is randomly selected and shown to the author, he can select a subset of questions to be replaced in a new iteration until the right sequence is generated.

Those steps done, the test can be activated and shared with other authors.

2.3. Didactics: Courses and Classes

The use of a shared repository of question and test allows a better and easier re-use of items: a tutor can have access, without restrictions, to a wide collection of resources to publish in the courses he administrates. There is a M:N relationship between tutor and course, that is a tutor administrates more courses and a course can be administrated by more tutors. The tutor manages the class: he can accept or deny learners' affiliation requests and expel a learner from the course.

Inside the course interface there are two different lists of tests presented to the learner: the "valuable" and the "self assessment" test list. Each test in the former list concurs to determine the learner's evaluation and it has an access control specified by a prerequisite expression and a maximum number of attempts. The latter list is just a guide for the learner to self train and assess: Each test in it is not influential in order to determine learner evaluation and it hasn't got any access restriction.

Adding a test to a list is an easy wizard-based operation: in the first screen the tutor must choose the test from a list of available tests and its presentation position in the valuable or self-assessment list. In case of adding a test to the "valuable" list, he has to specify access restrictions parameters as well. In the last screen, tutor must choose the evaluation strategy from a list, or create and instantly apply to the test a customized strategy, the evaluation scale and threshold. For a static test, there is an additional option to randomly shuffle the question order presented to the learner.

We believe that the choice to bind access restriction, evaluation strategy and other parameters concerning assessment, not to the test, but to the test presentation in a course test list, gives more flexibility to the application and more control of the class to the tutor.

Once the learner joins the test, a timer starts to measure the time he spent on that attempt. If he hasn't already done it before, the learner must deliver the test as the timer expires. After that, a summarizing table with test results is shown.

2.4. Prerequisites

Prerequisites are defined for each test in the "valuable" list. They establish, through a simple even powerful expression, the learner's right to access the test. If not fulfilled, prerequisites can deny learner's access to the test. Otherwise, provided that he didn't exceed the established maximum number of attempt, he can join the test. The only one language supported for the expression is *aicc_script*; a string expressed in such a language has a Boolean value and it is composed of the following elements:

- *Identifiers*: nouns that univocally identify a test in the valuable list.
- *Constants*: values that define the state of a test (*passed*, *completed*, *browsed*, *failed*, *not attempted*, *incomplete*).
- Logic, equality and inequality operators.
- A special syntax to define a set and to specify "at least n elements" from a set.

Example: the expression *test1 & 2*{test2, test3, test4}* is true if the state of test1 is passed or completed and at least two among test2, test3 e test4 are passed or completed.

A simple visual interface helps the tutor to define the prerequisites string without knowing aicc_script language. There is also an aicc_script-to-natural language translator to help the learner to better understand the prerequisites for a test.

A better and more complete explanation of aicc_script can be found in [SCORMCAM, 2001].

2.5 History and Report

A complete history of learner's performance on valuable test list is available to the tutor. Each record in the history contains the date and the time when the student has joined a test, the information whether in that moment the learner was inside or outside the university, the amount of time needed to finish the test and information about assessment (test score and state). There are two views of the same data:

1. Grouped by learner: tutor can have a snapshot of all learners' performance for each learner in his class.
2. Grouped by test: this kind of grouping is useful for the tutor to establish the effectiveness of the test.

A deeper inspection tells the tutor which response the learner gave to each question in the test. A very detailed history about every question is available as well. The tutor can easily establish the effectiveness of a question inspecting how many times it was selected to be presented in a test, the number and the percentage of correct, incorrect and not answered responses and the average time needed to get the response.

Another analysis of class and learner's performance is given by the report section. It consists of tables and charts that summarize the same data in the history in a more refined way: the results are shown by performance (best, worse and average) and by time (first and last).

First of all, the tutor can obtain the average performances of a learner on the valuable test sequence. A bar chart, that has the tests sequence on the X axis and the performance on the Y axis, shows the course of learner's performance. Since not all the tests of the sequence could have the same scale, performance is a normalized value on scale 100. For each test five performance values (best, worse, average, first and last) are shown.

The same analysis with the same kind of chart is available also for all the class: values shown are averages on all learners performance values. With this report the tutor can evaluate the intrinsic difficulty of a test or the average level of the class.

Another report shows the performances of each learner on the specified test. The chart shows the learner's names on the X axis and the performances on the Y axis.

Interesting information can be obtained from the macro-area based report. Here the tutor can get gap fillings and knowledge improvements with regard to a specified macro-area during the progress of the course. Contrary to previous reports, this one shows a multi-line chart, with the progress of time on the X axis (divided into weekly intervals) and the performances on the Y axis. Each line of the chart shows the performance on a specified macro-area. This report can show results both of a specified learner and of all class. Thanks to the information obtained from it, tutor can understand which subject should be better explained to the class, and which student has gap on what.

2.6. Assessment

As explained before, assessment isn't bound to the test, but to the association of a test to a course. eWorkbook provides a wide choice of predefined strategies and the chance to define a new customized strategy. Assessment is determined applying a function to some parameters, emphasizing more the one or the other according to the strategy's philosophy. Parameters taken into account are the following:

1. Response: correct, incorrect, no response
2. Number of questions in the test
3. Number of choices of the question
4. Weight of the question
5. Weight of the chosen response

In case of customized strategy, only the response parameter (the first in the list) is taken into account: tutor must specify the value to add or subtract to the total score in each of the three cases (correct, incorrect, no response). Whatever strategy is chosen, the final score must be reported in the selected scale (30 or 100). Predefined strategies are the following. The most of this list is taken from [IMSQTI, 2002a]. With 'score' we mean the assessment score before applying the scale proportion:

- *NumberCorrect*: score is determined as the number of correctly answered questions divided by the number of questions in the test.
- *NumberCorrectAttempted*: score is determined as the number of correctly answered questions divided by the number of questions in the test that have been attempted and not just selected and presented.
- *WeightedNumberCorrect*: score is determined as the sum of weights of the correctly answered questions divided by the sum of the weights of all questions in the test.
- *WeightedNumberCorrectAttempted*: score is determined as the sum of weights of the correctly answered questions divided by the sum of the weights of questions in the test that have been attempted

- *GuessingPenalty*: score is determined subtracting to the number of correctly answered questions the value 1/(number of choice of the question) for each incorrectly answered question.
- *WeightedAnswer*: score is determined as the sum of weights of the chosen answers divided by the sum of the maximum weighting answers for each question in the test.
- *L3*: score is increased by 1 for each correctly answered question, decreased by 1 incorrectly answered question. The total score is then divided by the number of questions in the test.

3. Architecture

As shown in the figure below, eWorkbook has a three-tier architecture like most J2EE applications. The tiers, or even components of the tiers, can be distributed across multiple hardware systems to improve system scalability and performance.

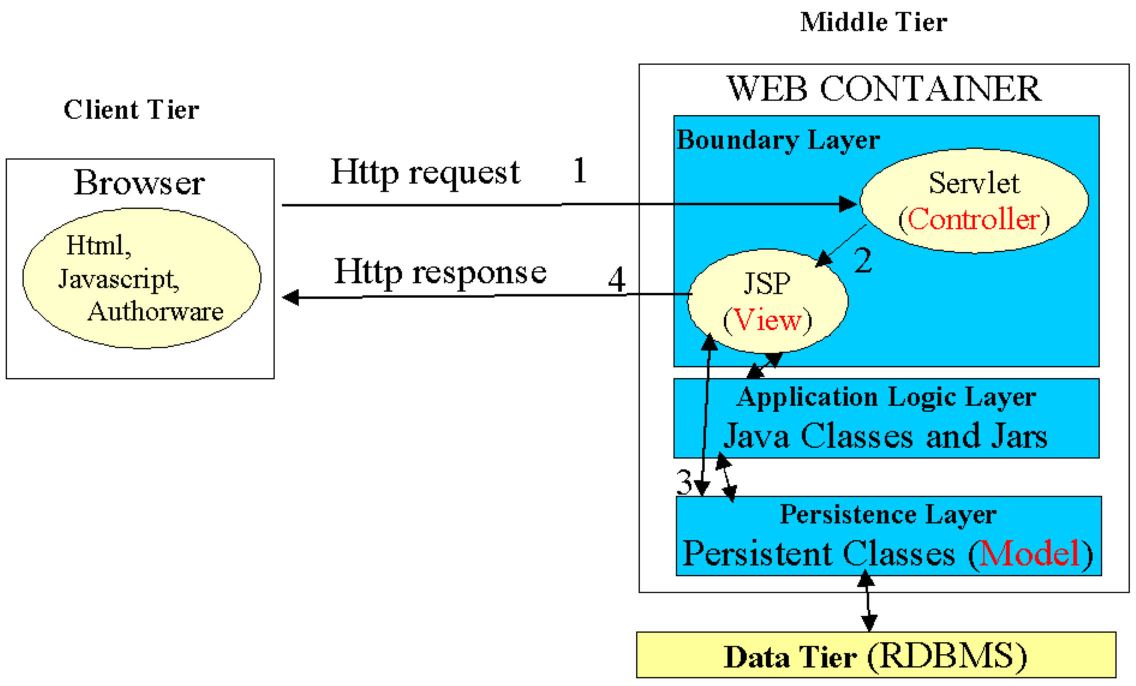

Figura 1 - eWorkbook Architecture

3.1 Client tier

As we said before, the application is fully accessible with a Web Browser. Navigation is easy thanks to the simple interface based on menus and navigation bars. User data inserting is done through HTML forms. Some form data integrity checks are performed through Javascript code. However, critical checks are performed both on the client side and on the server side. Macromedia Authorware plug-in is necessary for the learner to enjoy the tests. We chose to use it for its highly interactive interface and its simplicity in showing presentation features given the question data. The presence of the plug-in is not significant in terms of application security: it's just used for rendering purposes and no score computation is done on the client side. Instead, every user interaction is easily captured and sent to the server. All the client-server communication channel can be encrypted using SSL.

3.2 Middle tier

This tier contains the business logic of the application. It's further divided into three layers: Boundary Layer, Application Logic Layer and Persistence Layer. Our design implements the MVC Pattern, using its *model2* variant, suitable for Web applications. Its behaviour can be described listing the following steps:
All the HTTP requests are intercepted by a central servlet (the Controller)
The central servlet dispatches the control to the right JSP page (the View)
The JSP page communicates with the Application Logic Layer. This one uses the persistent classes at the Persistence Layer (the Model), instantiating new objects or loading existing ones, and querying, updating or destroying them. The passing through the Application Logic Layer can be skipped, that is the JSP page can use directly persistent objects.
The JSP page returns the HTTP response to the client.
The Apache Struts framework helped us in respecting this schema correctly.
The formerly described central servlet and JSP pages are Boundary Layer components. The JSP pages are primarily responsible for the interface rendering. This task is done through JSP statements and taglib elements embedded in HTML pages. To avoid the growing of code complexity, we chose to put just the code responsible for rendering in components at this level, leaving to the Application Logic Layer the biggest part of the computation. Some stand-alone units, like libraries and black-box modules were included as compiled jar files. All other components were based on Java classes.
The Persistence Layer at the middle tier is the nearest to the Data Layer. It is responsible to give an Object Oriented view of the data to the upper layers. Each relational entity type at the data tier matches a java persistent class at this layer. Relationships are mapped as references and collections. The Hibernate Framework was used to implement this layer. It requires the definition of some XML OO-Relational mapping files.

3.3 Data Tier

This tier provides data storage and other services to the application. It's accessed through the JDBC Java API, even though the Persistence Layer hides its use to the upper layers, giving them a higher level view based on persistent objects and a session object to manage them.

4. Third Party Frameworks

eWorkbook uses MacroMedia Authorware 7 for the rendering of test interface on the student side. Besides it is based on the following Open Source projects:

- Hibernate 1.2.5: for the OO-Relational mapping.
- Struts 1.2, to provide a *model2* architecture to the application. It also gives a big help in the form checking task and manages the connection pooling to the database.
- JCharts 0.7.0: A servlet-based library for chart generation.

5. Technical Notes

A big effort was profused to make the application running without limitation of operating system or proprietary software both on the client side and on the server side.

The application is distributed in the standard ‘J2EE war’ format, so it can be deployed on whatever J2EE server or Web container. Libraries and third-party frameworks are included in the distribution. DDL database script are only available for the Interbase-Firebird DBMS. An easy conversion should be performed in case of a different DBMS.

On the client side, only a common Web-browser is necessary to run the application. On the first time he/she gets into the test interface, the learner must install Macromedia Authorware 7 plug-in. It can be downloaded in a little time from the Internet at no cost for the end user.

6. Conclusion

Summarizing, the key-points of our proposal are the following:

- The resources authored with our tool are shared and highly re-usable
- The wide choice of assessment strategy and the possibility to extend that choice with new user-defined strategies, helps the tutor to tailor the test to the class’ competence and skill level
- The report section is rich of information and fit out of charts and tables. The tutor can have a complete and deep control over the performances of class and learners even on a single macro-area, and over the effectiveness of the authored resources. The learner can self-assess and fully get the benefits of blended learning.
- The definition of access rules, like prerequisites and attempt limitation, compels the learner to follow the right learning path.
- Our effort to make our application portable and usable makes it particularly suitable for the academic use for which it was thought for, even though it’s still a good choice in different ambits.

Even though multiple choice is the most common question type and it includes the true-false type, our team is working in order to support other types of question (*fill-in*, *matching*, *performance*, *sequencing*, *likert*, *numeric*, etc.) and questions based on external tools, like in [Leiva et al, 2000]. Other efforts are going to be made to introduce multimedia elements, like images, video and sound, in the rendering of the questions. An equation editor based on MathML [MathML] and an equation rendering engine would be a great feature to add.

It could be interesting to support import and export of question and test items in an IMS QTI [IMSQTI, 2002b] or QTI Lite [IMSQTILite, 2002] conformant format.

All the work done for eWorkbook project will flow together with other works into a wider project, aimed to implement a virtual learning environment framework, that will be used to integrate the features of an existing virtual learning platform.

References

[Blackboard] Blackboard, http://www.blackboard.com/

[Click2Learn] Click2Learn Aspen, http://home.click2learn.com/en/aspen/index.asp

[EduSystem] EduSystem, http://www.mtsystem.hu/edusystem/en/

[EduTools] EduTools, http://www.edutools.info/course/index.jsp

[Gusev and Armenski, 2002] Gusev M., Armenski G., onLine Learning and eTesting, 24th International Conference on Information Technology Interfaces ITI 2002, Cavtat, Croatia, 2002

[Hicks, 2002] Hicks C., Delivery And Assessment Issues Involved in Very Large Group Teaching, Printed and Published by IEE, London, 2002, pp 21/1-21/4

[IMSQTI, 2002a] IMS Global Learning Consortium, IMS Question & Test Interoperability: ASI Outcomes Processing, Final Specification Version 1.2, 2002

[IMSQTI, 2002b] IMS Global Learning Consortium, IMS Question & Test Interoperability: An Overview, Final Specification Version 1.2, 2002

[IMSQTILite, 2002] IMS Global Learning Consortium, IMS Question & Test Interoperability: QTILite Specification, Final Specification Version 1.2, 2002

[Leiva et al, 2000] Leiva W. D., Adriano C. M., Masiero P. C., Computer Supported Authoring of Questionnaires, ICECE'00, São Paulo, Brasil, 2000

[Lira et al, 1990] Lira P., Bronfman M. and Eyzaguirre J., Multitest II - A Program for the Generation, Correction and Analysis of Multiple Choice Tests, IEEE Transaction on Education, Vol. 33, No. 4, 1990, pp 320-325

[Lister and Jerram, 2001] Lister R., Jerram P., Design for web-based on-demand multiple choice exams using xml, IEEE International Conference on Advanced Learning Technologies, Madison Wisconsin, 2001, pp 383-384

[LManager] The Learning Manager, http://thelearningmanager.com/

[MathML] MathML, http://www.w3.org/Math/

[Proctiæ et al, 2001] J. Proctiæ, D. Bojiæ and I. Tartalja, test: Tools for Evaluation of Students' Tests - A Development Experience, 31st ASEE/IEEE Frontiers in Education Conference, Reno, NV, 2001

[SCORMCAM, 2001]. ADL (2001). The SCORM Content Aggregation Model, Version 1.2. Advanced Distributed Learning Initiative, 2001

[WebCT] WebCT, http://www.webct.com/