Novi Sad J. Math.
Vol. 38, No. 2, 2008, 115-131

MODULE FOR ONLINE ASSESSMENT IN AHYCO LEARNING MANAGEMENT SYSTEM

Ivica Botički[1], Ivan Budišćak[2], Nataša Hoić-Božić[3]

Abstract. The paper describes a module for online assessment in AHyCo (Adaptive Hypermedia Courseware) learning management system (LMS), developed at the Faculty of Electrical Engineering and Computing (FER), University of Zagreb.

A data model for knowledge assessment has been described, which includes defining questions and assessments. Along with standard, multiple choice questions, new programming questions for automatic evaluation of programming assignments or SQL queries are being introduced. A presentation of the system architecture and functionality of module for knowledge assessment has been given. This enables a new approach to creating, assigning and evaluating questions and assessments.

The paper discusses the usage of the described system in the course "Programming and Software Engineering" at FER for a continuous knowledge assessment introduced in the Bologna process of reforming the universities.

AMS Mathematics Subject Classification (2000): 68N01

Key words and phrases: technology supported assessment, learning management system AHyCo, e-learning

1. Introduction

One of the most important usage of e-learning systems in higher education is their ability to evaluate or asses the knowledge of a student. Online quizzes and assessments are among the most widely used types of evaluation with the two main purposes: they evaluate students' progress and help students in learning [4, 7, 11, 13]. During the past ten years the Department of Applied Computing of the Faculty of Electrical Engineering and Computing (FER), University of Zagreb was continuously enhancing student assessment methods, especially for courses with a high number of enrolled students (more than 700) in the 1st year of study: "Programming and Software Engineering" and "Algorithms and Data Structures" [2, 3, 10].

According to Bologna Declaration [1] FER, not only introduced the three main cycles of higher education, but also took this opportunity to completely

[1]Department of Applied Computing, Faculty of Electrical Engineering and Computing, Unska 3, Zagreb, Croatia, e-mail: ivica.boticki@fer.hr

[2]Department of Applied Computing, Faculty of Electrical Engineering and Computing, Unska 3, Zagreb, Croatia, e-mail: ivan.budiscak@fer.hr

[3]Department of Information Science, University of Rijeka, Omladinska 14, Rijeka, Croatia, e-mail: natasa.hoic@ri.t-com.hr

remodel the whole education process. Instead of a model where students were taking the lectures with irregular attendance and then repeatedly tried to pass the exams, the continuous assessment was introduced. The old assessment system, which was based on two mid-term exams or a classic written exam as an option, had to be upgraded. Due to the lack of teaching staff, improvements would hardly be possible without the introduction of a technology supported assessment system. Manual evaluation of homework and short tests was seen as technically impossible due to the large number of students and the lack of teaching staff [12]. Therefore, to support continuous student assessment, quizzes were introduced as a supplement to classic, paper-written exams [3, 14].

A Web-based LMS AHyCo (Adaptive Hypermedia Courseware) based on adaptive hypermedia was developed at FER in collaboration with the Department of Information Sciences, University of Rijeka, Croatia within several projects. Theoretical background of AHyCo and implementation of adaptive navigation are described in [8] and [9].

In the beginning, AHyCo has been used mainly for short online multiple choices tests in practical part of courses. The Bologna process adoption required additional test types in order to support midterm exams, homework and short tests. Although the development of a proprietary LMS might look unorthodox, in the year 2000, when the work on AHyCo was commenced, web-based adaptive systems almost did not exist. Furthermore, when continuous assessment was introduced, it appeared practical to have source code at hand in order to add numerous advanced features.

Besides functionalities included in most LMSs such as content authoring, delivering learning content and activities to students, student assessment, and discussion forums, AHyCo provides additional features such as the automatic evaluation of online programming assignments [6] and online SQL assignments, anonymous student surveying, administration of classes and groups of students, mid-exam administration and schedule planning, class attendance logging with smart cards and many more.

The rest of the paper describes the implementation of AHyCo assessment environment and discusses the technology supported assessment with AHyCo at FER.

2. Data Model and Object Model

Courseware in AHyCo is structured in lessons (concepts) grouped into modules. Modules can be grouped into courses. Knowledge assessment in AHyCo is conducted through assessments containing various types of questions. Each question is related to one or more lessons [8].

The existing data model for knowledge assessment was upgraded so it can ensure a simple and structured addition of different kinds of questions and testing, as well as special modules for scoring the tests.

The types of assessment questions are:

1. Classical questions – Multiple choice questions with one or more correct

answers. There are common hypermedia classical questions and parameterized classical questions [14].

2. Automatically evaluated SQL questions – Questions to which answers are SQL queries. Evaluation is done automatically through the module for automatic evaluation of SQL assignments [5].

3. Automatically evaluated programming questions – Questions to which the answer is a program written in a programming language. Evaluation is done automatically through the module for automatic evaluation of programming assignments [5].

To create an assessment the following parameters have to be determined: concepts with questions for the assessment, the number of questions in an assessment and configuration rules that determine the number of questions to be included in the assessment from each concept. Therefore, it is not necessary to include questions one by one since they are automatically and randomly generated prior to each new assessment. In addition to that, questions are permutated making plagiarism difficult.

In order to describe the data model, the ER model has been chosen. Schemes for questions, answers, assessment and assessment results performed within it are described in the remainder of the chapter.

2.1. Data Model: Questions and Answers

The data model scheme for questions ensures support for various kinds of questions (Fig. 1). All questions share some basic features, like question text, but also certain specific features related to the question type. For example, this way certain classical questions have predefined answers or function code for parameters (if the question text is parameterized), while programming questions and SQL questions have the answer text which is used for a comparison of assessment results during the scoring process.

Classical questions have two or more predefined answers of which one or more can be correct. A classical question can be parametric which means that the question text and/or the answer text can have parameters whose value is determined at the very moment of the assessment. If the classical question text contains parameters it is also necessary to define the script type, i.e. to determine whether the parameter values are determined using a function written in a script language *Visual Basic or Javascript* [10, 14].

Scoring automatically evaluated programming and SQL questions is conducted by input/output tests, i.e. by comparing the results acquired by performing the correct solution and student's solution. Programming and SQL questions, due to their specific scoring quality, demand a prefix and a suffix which will, during the testing process, be connected to the correct answer. For programming questions, the programming language needs to be defined as well (C or C++).

For testing programming questions, tests, which can be fixed and random, need to be defined. When working with fixed tests questions, fixed values of

input parameters and the expected output of software solution are defined When working with random tests, the value type is defined, as well as the upper and lower interval value from which input parameter values will be randomly generated for testing the software solution.

The data model for the answer needs to ensure the support for different question types. For classical questions, the answer permutations, as well as student's choice of answer need to be stored. For programming questions, the programming solution which the student created and the results of testing the solution need to be stored. The answer to the classical question needs to enable storing one or more chosen answers, because the classical question can have more than one correct answer. For programming questions, the result of running every test needs to be stored as well.

2.2. Data Model: Assessment and Assessment Results

The data model scheme for assessment had to ensure support for different assessment types. Each assessment type contains certain features like assessment name or assessment structure (units which compose the assessment), but also certain specific features related to assessment type. Classical question assessments have only one specific feature, and that is the maximum negative share per question, in order to calculate the negative number of points in the case of an incorrect answer. Negative points are introduced in order to avoid random guesses of correct answer by the students.

The realized number of points for the assessment is determined on the basis of realized number of points for each question in the assessment. Along with that, it is required to store statistic data on the number of correct, number of incorrect, number of partly correct, and the number of unanswered questions. These assessment results are associated with the specific student and the specific assessment.

2.3. Object model

Data stored in the relation database needs to be efficaciously approached and enabled performing various business processes which manipulate them. Because of that, business entities of a system are realized by objects, and the whole picture of the system is visible through an object model represented in this chapter.

The object model of the question has been taken as an example. The corresponding class diagrams exist for the answers to the questions, as well as for the assessments and assessment results.

Combining various kinds of questions in one assessment demands a transparent scoring on the level of question. Questions should be able to "score itself" depending on the type, and store the student's answer. This problem can be solved by inheritance in object oriented design. It is necessary to define an abstract class *Question* and corresponding abstract methods *Score* and *StoreStudentAnswer* which will, in inherited classes, perform question scoring and storing the answer. Each of the inherited classes (for each question type a separate class which inherits class *Question* has been created) will override

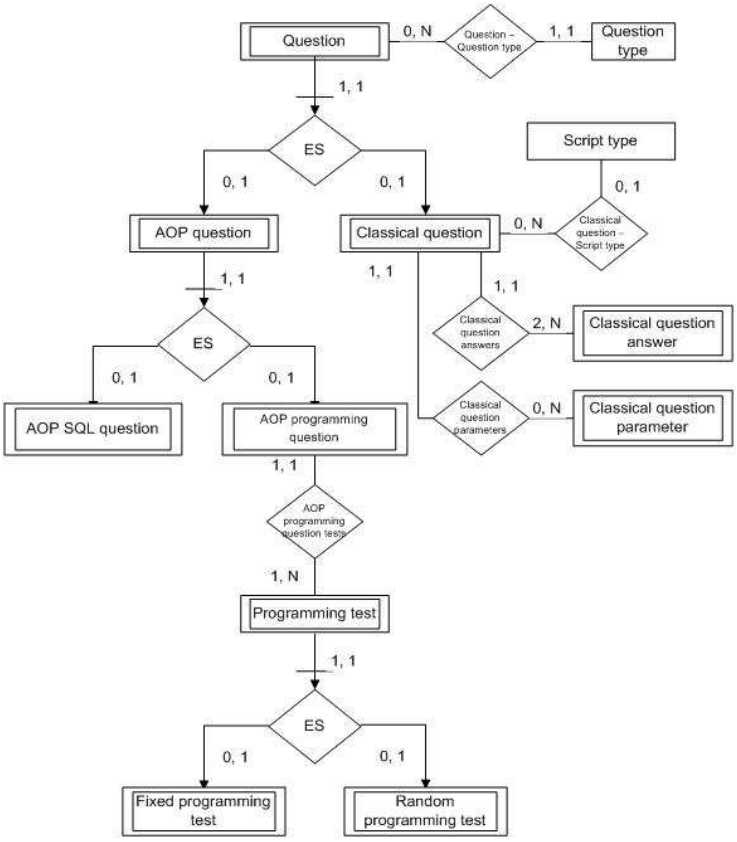

Figure 1: An example of ER scheme for question

methods *Score* and *StoreStudentAnswer* and implement its logic of scoring and storing the answer.

Automatically evaluated questions will additionally have to implement a method *CheckSyntax*, which will perform syntax check of student's solution. Since syntax check is typical for automatically evaluated questions only, and abstract class *AOPQuestion* has been defined. This class contains an abstract method *CheckSyntax* which all inherited classes must override.

Since classes belonging to business logic layer represent abstract entities which can map to more database entities, it is necessary to implement methods for storing, modification and deletion of these abstract entities. Toward this end, an interface IbusinessObjectActions has been defined, and it defines methods *Insert, Update* and *Delete.* Each class needs to implement this interface in order to enable the classes of user interface layer to perform certain mentioned operations.

A class diagram, which illustrates an object model of a question, is shown in Figure 2.

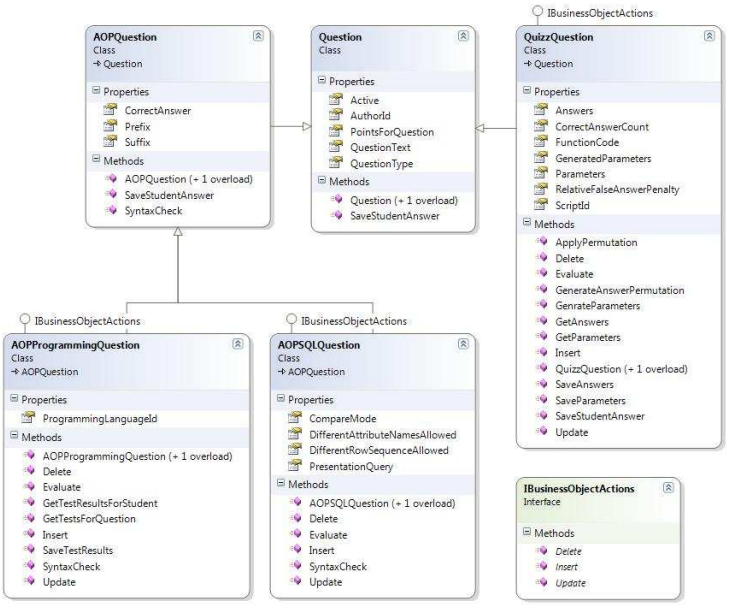

Figure 2: An example for object model: class diagram for questions

3. System Architecture

Student knowledge assessment module is a part of the learning management system AHyCo. AHyCo is a web-based learning management system which can be accessed using a web browser through the Internet or intranet. AHyCo is implemented in Microsoft's ASP.NET technology, consists of a database server, runs on a web server and is accessed by Web browser.

Knowledge assessment module is based on a three-tiered architecture (Fig. 3) with Data Access Layer, Business Logic Layer, User interface layer, and Presentation layer [5].

Data access layer is realized with Microsoft's ADO.NET technology that enables simple integration with a database management system, namely SQL Server. Namespace ProvjereLib.DataAccess contains classes for storing, retrieval, modification and deletion of data. The main idea behind the data access layer is that it does not depend on the higher, more abstract layers, i.e. business logic layer. All class methods in the namespace ProvjereLib.DataAccess depend only on the methods provided by ADO.NET.

The business logic layer (BLL) has been modeled on the basis of already described ER data model. BLL models all entities participating in the module's business logic. Its classes are stored in the ProvjereLib.BusinessObjects namespace. The main idea is to provide an abstraction from the data access layer which deals with primitive data operations. BLL's classes therefore model assignment evaluation, question test case evaluation, validate input data, etc.

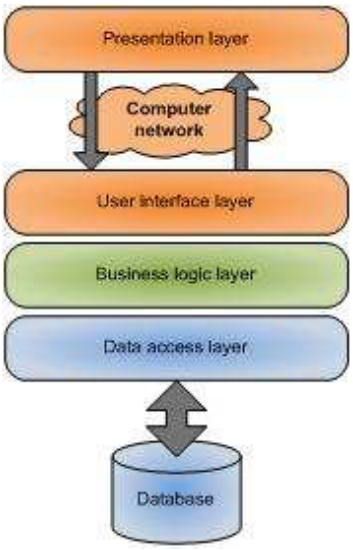

Figure 3: Three-tiered architecture of AHyCo's module for knowledge assessment

The UI layer runs on the server side and its classes are stored in the ProvjereLib.GUIObjects namespace. It communicates with the business logic layer therefore connecting presentational layer with the business entities. Web application's user interface represents module's presentation layer providing simple validation prior to the server request processing [5].

4. Knowledge Assessment Modules

Learning management system enables input of questions through the *question module.* It is possible to enter any of afore-mentioned question types, and the input is performed through a web interface. The question module enables modifications on already stored questions, these being question text modifications, answer text modifications (if the question has predefined answers) and other question parameters.

Defining assessment and preview of the performed assessments can be performed using *assessment module.* Since questions can be of a different type, it was necessary to implement other assessment types. The assessment type determines question type which can be found in the assessment. An assessment can have only one question type, and this is how classical assessment, programming assessment and SQL assessment differ. Combined assessments are those assessments which, alongside classical questions, have also programming and SQL questions.

Assessment preview can show and, if needed, modify student answer. If there was a need to modify the student's answer (e.g. due to technical difficulties in

system performance), the assessment can be evaluated again.

Finally, the results of realized assessment can be seen by the student on a webpage for assessment results. The student is shown a list of all questions in the assessment, with realized number of points for each question, and the total realized number of points in the assessment. The remainder of the chapter describes the question and assessment module.

4.1. Question Module and Question Management

A special webpage for question preview enables the representation of all questions belonging to a certain lesson, and easy access to the webpage for modification and entering new questions. The table shows question identifier, question text, question type, question author and whether the question is active (i.e. whether it is used in assessments). The table also has links to a webpage for question modification and a link for question deletion.

At the question entry, it is essential that the text editor enables picture entry, table view, text formatting, etc. There are many controls which enable text editing within a webpage, however, none of them offers as many options as text editors available in the form of desktop applications. .NET technology enabled implementation of control by which text can be edited using Microsoft Word text editor. It is possible to run Microsoft Word via the web application on a local computer and store the content of Word document in HTML form. This enables complex text entry, and since it is stored in HTML form, it is simple to display it on a webpage. In order to satisfy users who are not able or do not wish to use Microsoft Word text editors, text entry using HTML text editor is also enabled.

On the page for classical question entry, alongside question and answer text, it is also possible to define a lesson which the question belongs to, and question parameters if we are dealing with a parameter question.

A parameter question enables parameter definition which can be in the question text and/or in question answer. Parameters are added into the question text as tokens in special format -§ParameterName§. During question preview, parameter values are determined dynamically using function code in one of the script languages *Visual Basic* or *Javascript*. Also, tokens previously entered into the question text, i.e. answer text are replaced.

Programming question entry can be divided into two parts: question text entry with correct answer and test entry for the question.

The question text, which will be shown to the student, is entered by Microsoft Word text editor. The next step is entering the correct assignment solution which consists of three parts: prefix, correct answer and suffix.

Field *Correct answer* contains correct assignment solution provided by the teacher, while the *Sufix* field is used in assignments that require a function implementation from students. In the case of programming assignments in programming language C, this field is used to define the main() function that reads input test parameters from the standard input using scanf() function, calls the function specified in the assignment (one that students submit as a solution)

and outputs the results by using the printf() function. Figure 4 illustrates an entry page.

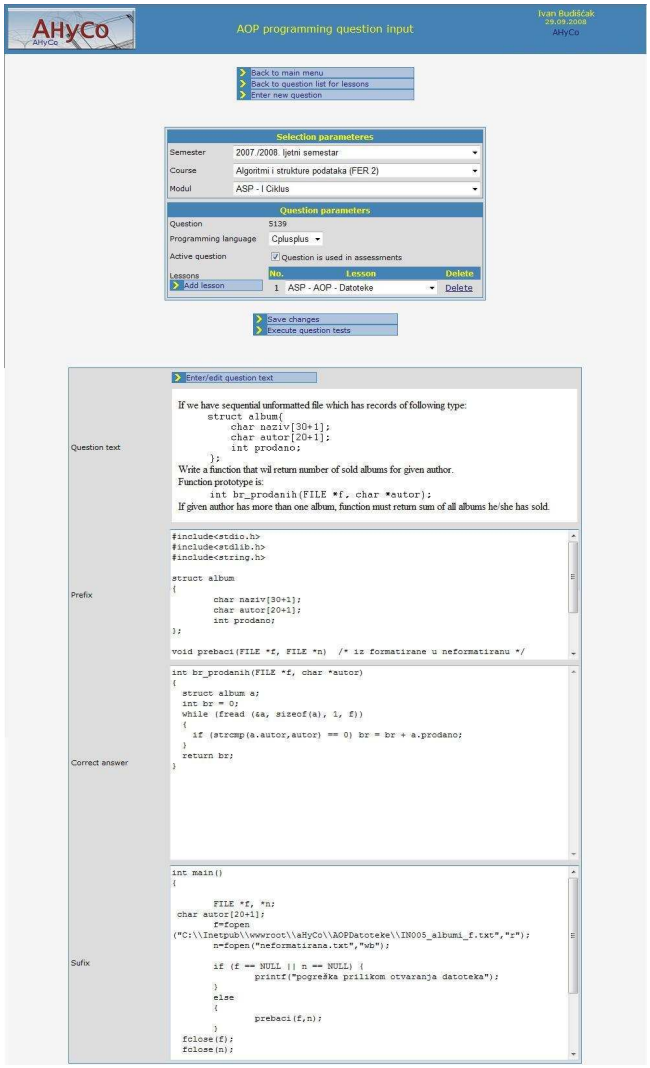

Figure 4: Programming question entry

For each test, it is possible to define its relative share in question scoring. A question is scored in a way that the maximum number of points is distributed depending on the shares of tests. It is possible to define two test types: fixed and random tests. After the question and the correct answer have been entered and tests defined, it is possible to run a trial of the test performance.

SQL question is alike programming question, and question text is entered using HTML editor or Word. Besides question text, the correct answer to the

question is defined, the testing suffix which checks SQL query results during entry, and a query which returns results for presentation to the student (Fig. 5). The SQL query is run in a special database which was created for the needs of performing tests with automatic scoring. Module that performs automatic testing of SQL assignments is created in such a way as to be able to connect to any other database.

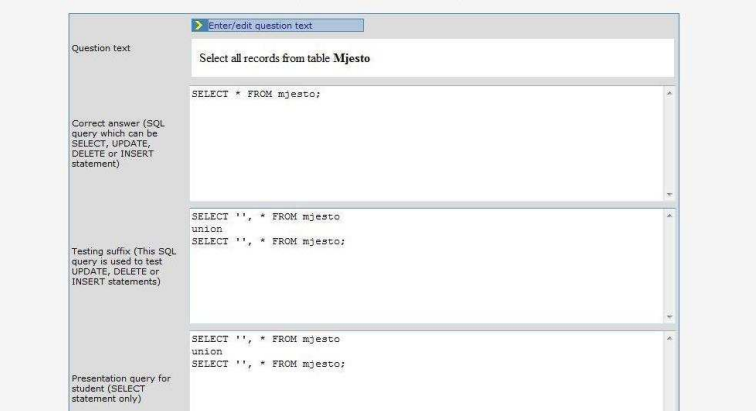

Figure 5: SQL question entry

4.2. Assessment Module and Assessment Management

Defining assessment, creating assessment and assessment preview can all be performed via module for assessment management. It is possible to preview and modify parameters of existing assessments or define a new assessment on the webpage for assessment definition. Web form for assessment preview is designed generically and uses the question object model in order to publish the scoring of any type of question. For every question type it is possible to preview the answer, i.e. student's solution. It is also worth mentioning that the same web form has been used for assessment preview as for creating assessment.

Assessment definition consists of two parts: assessment parameters' definition and assessment composition selection. After the assessment has been defined by entering the type, name, password, number of question, points and other basic data, it is necessary to make a choice of assessment composition, i.e. the minimum and maximum number of questions from each selected lesson.

Created assessment starts by entering the password into the corresponding field on the main page of AHyCo system.

Web form for creating assessment can show any type of question. Depending on the question type, the webpage preview changes, and by that, functionalities specific for certain question type also change. Additionally, web form for assessment preview can display questions in three ways of operating: taking assessment, assessment result preview for students and assessment result preview for teacher. In assessment result preview for teacher it is possible to reevaluate

the assessment, while this is not the case in the assessment result preview for students. Figure 6 illustrates the web form with an SQL question during the process of creating assessment.

Figure 6: SQL question in assessment

During taking assessment, the webpage shows a time counter which counts off the remaining time for taking the assessment. The assessment can be submitted for scoring at any point in time by clicking the link *Score Assessment*. The student is first being redirected to a page which shows their answers (for classical questions) or syntax accuracy (for automatically evaluated questions). Here, the student has the possibility to continue with the assessment or submit the assessment for scoring. By submitting the assessment, the assessment is considered finished and it is no longer possible to go back to it. After the scoring, a webpage appears showing the summary results of the assessment.

By visiting the webpage for previewing realized assessments, the teacher can see the list of realized assessments for each defined assessment. Realized assessments are shown in a table, and there are also links for assessment preview, assessment deletion and enabling assessment taking continuation. Also, statistic data on the score is shown as well as data on the number of correct answers, incorrect answers, partly correct answers, and the number of unanswered questions.

Figure 7 illustrates assessment results preview for SQL question. The teacher can see student's solution, the correct solution and the result of student's solution performance.

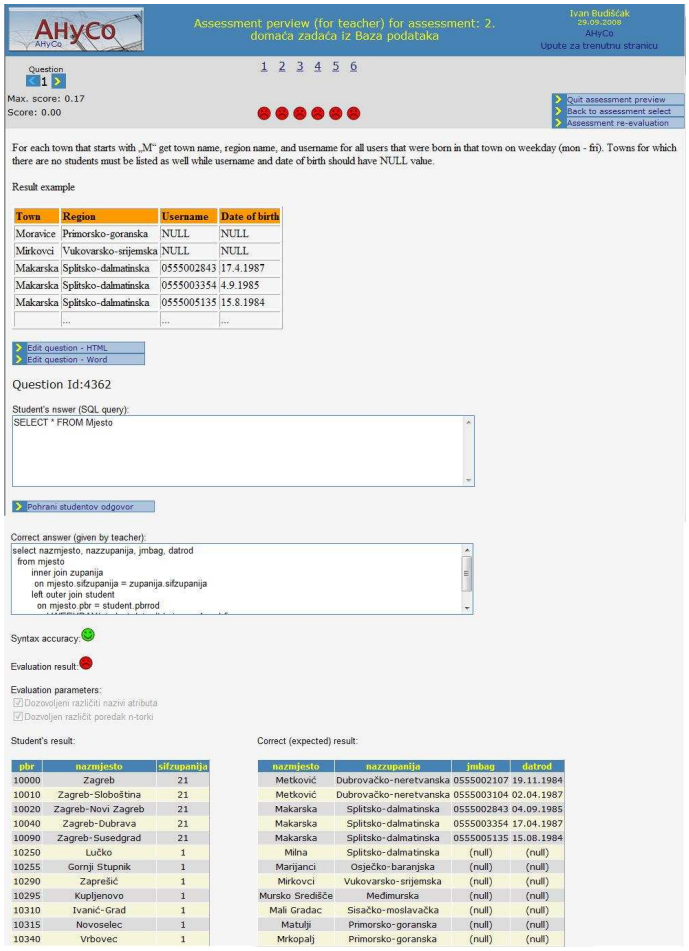

Figure 7: Assessment preview for teachers – SQL question

5. Assessment Methods for "Programming and Software Engineering" Course

According to the Bologna declaration [1] all courses at FER are graded through continuous assessment, consisting of homework, periodical tests, two mid-term exams and the final exam. All student activities contribute with a certain number of points. Maximal number of points earned on a course is 100%.

The students will achieve final grades after the final exam. Minimal number of points necessary to pass the exams is 50%. A student will fail an exam if the total number of collected points is less than required minimum. In that case the result of the first final exam is deleted, and the student can once repeat the final exam.

The students above the 50% are given the grades based on the normal distribution. The final grade depends on the position on the rank-list of total points, as follows:

- grade 5 (A) – 15 % of students
- grade 4 (B) – 35 % of students
- grade 3 (C) – 35 % of students
- grade 2 (D) – 15 % of students

The final score for the course "Programming and Software Engineering" is composed out of 70% gained from mid-term exams and final exam, 6% from homework, 18% from quizzes and 6% from classroom activity. Mid-term exams and final exam are performed in a classroom, the classic way, because the aim is to ensure objectivity by giving all students the same assignments. Due to a large number of students (over 800), it is not possible to ensure that everyone takes a test on a computer at the same time, in a controlled environment under the supervision of a teacher [3].

Other activities (homework, quizzes) which are performed during the semester are implemented as online AHyCo assessment with online multiple choices tests and programming assignments. Performance assessment of knowledge is realized as homework in the form of programming assignment. Students upload homework and get immediate feedback from the system on the achieved results. In 2007/2008 students took 6 online quizzes and 3 homeworks, and due to the large number of students (in 2007/2008, 868 students took the course) such a continuous assessment included over 5000 multiple choice questions and over 2500 programming assignments. These numbers show that without the help of AHyCo system with only 6 teaching assistants which run the practical part of the course, test and homework evaluation and provide feedback to the students would be close to impossible.

With the help of AHyCo system, teachers can more easily track course statistics due to the fact that the results of mid-term and final exams are also entered into AHyCo. It is therefore possible to retrieve data on student's points for every taken assessment and the whole course. Also, data on student's success achieved in previous academic years is also stored, so it is possible to follow and compare student's success through the course of a few years and by certain assessment methods. The analysis of the results in the past 3 years has shown that there was a relatively high success rate on the exams, over 65% (this is especially important because we are dealing with a large number of students, around 800) right after the end of the semester, at the first final exam.

6. Related Work

Computer supported assessment has been used since the early days of technology supported learning. It should motivate students to learn better and

make teachers' job easier by reducing knowledge assessment time for large number of students. Tools for student assessment have been included in many LMSs since their inception [15, 16, 17]. Nevertheless, only simple question types like true/false questions, multiple choice questions, questions with short answers, matching questions and upload of non-automatically evaluated essays were supported. More complex systems for automated evaluation of programming and SQL assignments are still not standard parts of most LMS's, including the prevailing ones like Moodle [15], Sakai [16], WebCT and Blackboard [17]. On the other hand, many independent systems for either automatic evaluation of programming assignments or for automatic evaluation of SQL assignments exist today.

Intelligent tutoring system (ITS) SQL Tutor and its more recent version SQL Web [21] are considered to be older examples of teaching SQL. They are focused on personalized tutoring and teaching, not on the student assesment. More up-to-date systems for automatic assessments of SQL are Corrector [22] and LEARN-SQL [23], which use real databases and are not exclusively limited to SELECT statements.

Although most systems for automatic evaluation of programming assignments deal with one specific programming languages, there are a few examples that use more programming languages such as Online Judge [6] (C, C++, Java). SISA-EMU [18] is used to evaluate assembler programs, Automatic Marker [19] for Java and ShareMe [12] for network programming and distributed systems. SAW [20] is an LMS that incorporates learning modules for geometry and programming while AulaWeb [7] has a module for automated grading of TurboPascal assignments.

There is work in progress to include automated grading as independent modules into different LMSs. LEARN-SQL [23] can be used within Moodle and Automatic Marker within Sakai [19]. SISA-EMU [18] can be used within Moodle as a system for learning assembler programming primarily which gives error feedback to students and does not provide automatic grading services.

Module for automated grading is a component of AHyCo LMS and is therefore integrated with other parts of the system. It gives students opportunity to evaluate their knowledge while teachers have the possibility to browse results of automatically graded assignments. AHyCo's authoring component is used to define automatically graded tests and questions together with the test cases for programming and SQL assignments. By utilizing its object programming model, module functionality can be reused and assignments in different programming languages (currently C, C++, C#, SQL) can be supported . Additional languages can be easily added by creating language-specific providers.

7. Conclusion

This paper has presented the assessment module of AHyCo LMS as well as enhancing assessment mechanisms in the course "Programming and Software Engineering" at the Faculty of Electrical Engineering and Computing (FER).

A continuous knowledge assessment of a large number of students introduced with the Bologna declaration at FER would be virtually impossible without using ICT, i.e. a system for online knowledge assessment. Because of an unfavorable ratio of teacher and students, teacher spend a significant amount of time on scoring student's assignments taken the standard way, on a piece of paper, instead of investing their time on improving the teaching process. AHyCo system significantly disburdens the teacher because most of the assessment is implemented with the help of LMS in the form of quizzes or short online multiple choice tests.

It is important to enable students of engineering and computing for programming and working with databases. Due to that, it is necessary to enable online testing of programming solutions and SQL queries, which still has not been implemented in most LMSs, and represents a significant novelty of AHyCo LMS. For the knowledge assessment module of AHyCo learning management system, a new approach to the process of creating, assigning and evaluating online assessments has been developed, with the emphasis on automatic evaluating of programming assignments.

The system is currently being used at the Faculty of Electrical Engineering and Computing, University of Zagreb, and it is being constantly improved, according to student and teacher demands. Future plans of developing AHyCo module for evaluating programming assignments include system improvement based on the analysis of data on student results. Also, module expansion with new programming methods is planned, testing mechanism upgrade and using mobile devices which will further improve ways of student assessment.

References

[1] The Bologna Process, 2008. `http://public.mzos.hr/Default.aspx?sec=2519` (last accessed June 15, 2008).

[2] Botički, I., Pukljak-Zoković, D., Nizetić, I., The analysis of student activity by the automatic evaluation of programming assignments in an online learning environment. Proceedings of the ICAT 2007 Conference, Sarajevo, Bosnia and Herzegovina, Oct 29-31, 2007, CD-ROM edition.

[3] Botički, I., Milašinović, B., Knowledge Assessment at the Faculty of Electrical Engineering and Computing. Proceedings of the ITI 2008 Conference, Cavtat, Croatia, June 23-26, 2008 (in print).

[4] Brusilovsky, P., Miller, P., Web-based Testing for Distance Education. Proceedings of WebNet 99, Honolulu, Hawaii, United States, 1999.

[5] Hoić-Božić, N. Budišćak, I., Botički, I., Online Assesment of Programming Assignments in a Learning Management System AHyCo. Engineering Review Vol. 28 No. 1 (2008), 51-56.

[6] Cheang, B., Kurnia, A., Lim, A., Oon W., On automated grading of programming assignments in an academic institution. Computers and Education, Vol. 41 No. 2 (2003), 121-131.

[7] Garcia-Beltran, A., Martinez, R., Web Assisted Self-assessment in Computer Programming Learning Using AulaWeb. International Journal of Engenering Education Vol. 22 No. 5 (2006), 1063-1069.

[8] Hoić-Božić, N., Mornar V., AHyCo: a Web-Based Adaptive Hypermedia Courseware System. Journal of Computing and Information Technology – CIT Vol. 13 No. 3 (2005), 165-176

[9] Hoić-Božić, N., Mornar, V., Authoring of Adaptive Hypermedia Courseware Using AHyCo System. Advances in Web-Based Education: Personalized Learning Environments, (G.D. Magoulas, S.Y. Chen eds.), pp. 253-276. Hershey: Idea Group 2005.

[10] Hoić-Božić, N., Mornar, V., Pukljak Zoković, D., The Model for Testing in Adaptive Hypermedia Courseware. Proceedings of the ITI 2003 Conference, Cavtat, Croatia, June 16-19, 2003, 255-260.

[11] Jones, E., Grading student programs - a software testing approach. Proceedings of the fourteenth annual consortium on Small Colleges Southeastern conference, Roanoke College, Salem, Virginia, United States, 2000, 185 – 192.

[12] Kerer, C. et al., ShareMe: Running a Distributed Systems Lab for 600 Students With Three Faculty Members. IEEE Transactions on Education Vol. 48 No. 3 (2005), 430-437.

[13] Kurt, A., Kubat, C, Oztemel, E., Web-Based Virtual Testing and Learning in Material Science and Engineering. International Journal of Engenering Education Vol. 22 No. 5 (2006), 986-992.

[14] Mornar, V., Hoić-Božić, N., Pukljak Zoković, D., Approaches to Online Testing in Web-based Educational Systems. Proceedings of EUROCON 2003, Ljubljana, Slovenia, September 22-24, 2003, 343-346.

[15] Moodle. `http://www.moodle.org`, 2008.

[16] Sakai. `http://sakaiproject.org/portal`, 2008.

[17] WebCT, BlackBoard. `http://www.blackboard.com/`, 2008.

[18] Jimenez-Gonzalez, D. et al., Work in Progress–Improving Feedback Using an Automatic Assessment Tool. Paper accepted for 38th ASEE/IEEE Frontiers in Education Conference, Saratoga Springs, NY, October 22 – 25, 2008. `http://fie.engrng.pitt.edu/fie2008/papers/1667.pdf`

[19] Suleman, H., Automatic Marking with Sakai. Paper accepted for Annual Conference of South African Institute of Computer Scientists and Information Technologists (SAICSIT), Wilderness, South Africa, October 6 - 8 2008.

[20] Moura, J., Brandao, L., Brandao, A., A Web-Based Learning Management System with Automatic Assessment Resources, 37th ASEE/IEEE Frontiers in Education Conference, Milwaukee, WI, October 10 - 13, 2007.

[21] Mitrović, A., Hausler, K., Porting SQL-Tutor to the Web. ITS'2000 workshop on Adaptive and Intelligent Web-based Education Systems, Montreal, Canada, June 19-23, 2000, 37-44.

[22] Coelho, F., Corrector, a web interface for practice sessions. Proceedings of the TICE 2006 "Information and Communication Technologies in Higher Education and Industry", Toulouse, France, 25-27 October 2006.

[23] Abello, A. et. al., LEARN-SQL: Automatic Assessment of SQL Based on IMS QTI Specification. Proceedings of the 8^{th} IEEE International Conference on Advanced Learning Technologies, Santander, Cantabria, Spain, July 1-5, 2008, 592-593.

Received by the editors July 10, 2008