A Web-based Self-assessment System with Multi-language Programming Questions

ÁNGEL GARCÍA-BELTRÁN, SANTIAGO TAPIA, RAQUEL MARTÍNEZ, MANUEL GONZÁLEZ
Department of Automática, Ingeniería Electrónica e Informática Industrial
Universidad Politécnica de Madrid
ETSI Industriales, C/José Gutiérrez Abascal, 2. 28006 - MADRID
SPAIN

Abstract: - This paper describes a web assisted self-assessment application with multi-language programming questions. The whole user-system interactivity, based on a client-server architecture, is carried out by means of a computer connected to the Internet with a web browser. Practical matters of authoring code questions and their implementation and use methodology are explained. These self-assessment code questions are meant not only to encourage and motivate the students but also to assess them. Finally, the results of a survey of students' perceptions and the influence of these on future developments are presented.

Key-Words: e-Learning, b-Learning, self-assessment, programming questions

1 Introduction

The purpose of this work is to present the design and implementation of a web-based self-assessment environment with multi-language programming questions.

This application has been implemented in a complete e-learning system, named AulaWeb [1], and is being used as a facility to encourage students to practice programming techniques in computer science courses with different programming languages, for example, TurboPascal, Java and C/C++. Furthermore, this paper describes the pedagogical methodology and some results drawn from this experience.

AulaWeb system has been used as an on-line support of courses by more than ten thousand students of the Universidad Politécnica of Madrid since 1999.

AulaWeb exploitation consists of four main activities in more than three hundred courses: theoretical and practical content, open discussion forums, self-assessment exercises and homework delivery. Each tutor can exploit these activities depending on the course methodology and its characteristics.

2 Self-assessment

In this way, self-assessment may appear as a key activity in courses with an enormous number of students (i.e. Computer Science courses). A major challenge for tutors is encouraging the students to engage actively in learning programming fundamentals, and, in this way, a regular assessment system is essential.

2.1 Reasons for a Regular Testing

A regular assessment forces the students to increase their motivation, study harder and practice the material actively. These tests are used to improve the performance of the students, focus their activities and so drive them to practice computer programming during the academic term.

Furthermore it provides an opportunity for the professor to get frequent feedback about how well students are learning and assimilating the contents of the course [2]. Besides, both the students and the tutor know much earlier if the material is not being understood. To be effective, feedback should be given immediately after the student has finished the test [3]. The main drawback: a considerable amount of time may be required to prepare, grade and implement these frequent exercises.

With large groups of students, it would be especially impossible to implement this kind of assessment without computers. There are many authors that report about different computer and web assisted assessment tools to develop a regular testing system [4]–[9], but none of them is specifically focused to students of programming languages.

[1] This work is funded by the DGES (Ministerio de Educación y Ciencia of Spain) under contract SEJ2004-08004-C02-02.

2.2 The Self-assessment Module in AulaWeb

There are many types of questions (single-choice, multiple-choice, short answer, true-false, etc) implemented in AulaWeb platform [10]. However, in a programming course, tests should be driven preferentially to questions with code answers. In fact, from the 2001/02 academic year onwards, the environment has been specifically made suitable for self-assessing computer programming skills in TurboPascal [11]. The feedback from this use has resulted in improved systems and methodologies incorporating new implementations, great experience and best practices.

The self-assessment module is based on a questions database, with a friendly and easy-to-use interface for adding and updating questions. Tutors can configure exercises indicating the quantity, the level of difficulty, the type and the syllabus chapters of the questions. When finishing the exercise, results are stored in the database and the system allows the student the possibility of checking his exercise and comparing his/her answers to the correct solutions. Evaluation of the exercise is, therefore, automatic, and the student and his/her teacher can access the results of the self-assessment activities.

The following section explains how the multi-programming code questions module has been developed in AulaWeb.

3 Code Questions

Code questions are a new development of the AulaWeb platform oriented to any programming language learning-teaching. Aim: the system asks the student a code-type question in order to carry out an specific task and subsequently correct his answer.

3.1 Preparation

The implementation of each code question involves the following elements: a wording, a set of code files with gaps to be completed by the students and a set of code files used to check the students answers.

3.1.1 Wording

The wording indicates what task the code must do. For example: *Complete, compile and run the following Java code in order to calculate the square of a real value …*

3.1.2 Code Files Attached to the Wording

Questions may incorporate several code files (one of them must implement the main method of the application) with none, one or more uncompleted gaps to be fulfilled by the students. In the most basic case there is only one code file with the main method and one or more gaps. The gap positions are marked in the code file with a pair of specific code comments: "`//##user code##`".

Once the student has filled them and click over the Compile option, the answers are sent to server and the complete code files are compiled. Subsequently the server gives back the result of the compilation.

The following code is a very simple example of an alone code file attached to the wording:

```
/**
 *  SquareTest
 *  Purpose: Square method test
 *  for Java Programming
 *  Author: A. Garcia-Beltran
 *  Date: March 4th, 2006
 */
public class SquareTest {
  public static void main (String [] args) {
    System.out.println("The square of 7.5 is: "
                       + square(7.5));
  }
  public static double square (double x) {
    //##user code##

    //##user code##
  }
}
```

3.1.2 Code Files for the Correction

These files are not provided to the students, but they are used by the server to collect the student answers and check them. These code files have to include the corresponding gaps for the answers and some routines to verify that the answers will carry out the wording task properly using any technique for software testing, for example, a black-box method. Once the student has filled them and click over the *Run* option, the answers are sent and these checking code files (completed with the corresponding student's codes) are compiled and executed in the server. Later, the server returns the result of the execution. The following code is the code file for checking corresponding to the previous example:

```
/*
 * SquareTestCorrector: to check the answer
 */
public class SquareTestCorrector {
  public static void main(String[] args) {
    System.out.println(functionalChecking());
  }
  public static double square(double x) {
    //##user code##

    //##user code##
  }
  public static String functionalChecking() {
    double x = Math.random()+3;
    double aux = x*x;
    boolean t = Math.abs(aux - square(x))<=0.001;
    for (int i=0; i<5; i++) {
      x = x - 1;
      aux = x*x;
      t = t && Math.abs(aux - square(x))<=0.001;
    }
    if (t) { return ("The result is CORRECT"); }
```

```
        else return ("The result is INCORRECT");
    }
}
```

3.2 Student Interface

Figure 1 shows the interface of a code question which includes a virtual typical programming environment implemented by means of the Java applet. This interface has been built using the ASP technology of AulaWeb [12] and a Java applet that emulates a Virtual Programming Environment and shows the two first elements described in the prior section. Students can read the wording (HTML) and use the edition window of the virtual environment to input and edit the answer codes in order to complete the whole application. The corresponding gap positions are marked in each code file in the edition window with a pair of specific code comments "`//##user code##`".). In the question example there are two code files (with the corresponding tab) and only two gaps in the first code file and the correct answers are `return (height*width);` and `return (this.area()>b.area());`, respectively.

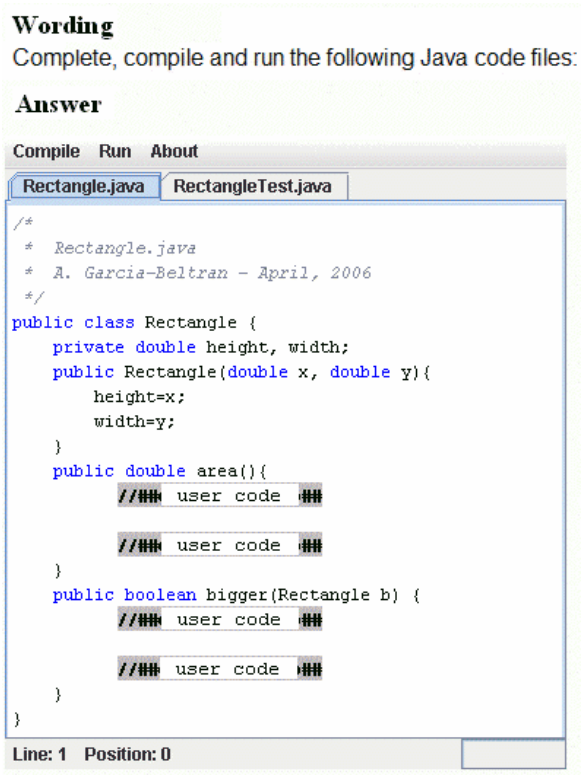

Fig. 1. Example of a question interface

The environment also includes a menu bar with options to compile and run the complete application. The *Compile* option sends the answers to the server and the server compiles the code from the edition window. For this reason, an on-line compiler has been installed in the server in order to support this automatic checking for the Java code answers. When compiling is complete, a status window appears. If a compile-time or syntax error occurs, an error message is shown like a real programming environment. For instance, if the student tries to compile the first code file with the answer: `retun (height*width);` for the question example, then the system displays the corresponding message window. If the student answers have no compilation errors, i.e.: `return (2*height*width);` then the window indicates a message of *Successful Compilation*.

The *Run* option executes the input code together with a checking model code in order to detect run-time or logical errors. In both cases, the corresponding error message is shown. For the self-assessment in logical errors, student code outputs are compared with randomly generated model outputs in the server. For instance, if the student tries to run the application with the code answer: `return (2*height*width);` for the question example, then the system displays the corresponding ("*the result is wrong*") message window. Each specific mistake may require the display of a customized error message previously written in the code file for checking. If the answers have no logical or run-time errors, then the window indicates a message of *Successful Execution*.

3.3 Teacher Interface

The system makes easy the introduction of a new code question in the database by means of a step-by-step assistant. Authors must insert a short description, the corresponding syllabus unit, the wording, the code files attached to the wording, the code files for checking and the theoretical difficulty level. A significant amount of time is required to prepare one code question, but once a set of questions have been accumulated, they can be recycled in later courses in the future. Teachers can also use the same code question interface that the student can see and reports windows about the results of the students' exercises and the questions database. So, the system allows the teacher to track the student's progress during the course and also provides some statistical tools to compare the theoretical and experimental level of difficulty of the questions and to revise the first ones.

3.3 Multilanguage Programming Questions

The above development is intended for Java programming language only, but a new improvement has been implemented to meet more demanding functionalities. This application can deal with more complex questions including multi-programming language, since the compiling and running process

for testing the students' answers are fully configurable by means of a *makefile*. The student's interface is the same, but the question design changes. A new task of the question design is writing the instructions for compiling and executing the students' answers. As before, the system looks for the students' answers and inserts them in the corresponding gaps of the testing program. Then it looks for the *makefile* and executes the GNU make application [13] with options "compile" and "all". In doing so, the answers should be compiled and their functionality checked with the appropriate commands. As any command can be used in the *makefile*, the teacher can choose among a wide-variety of tools or, even more, design his own tools. Up to date, questions for Java and C/C++ programming languages have been designed and implemented. In the future, we are working to develop questions for MatLab, FORTRAN, Maple and SQL. There are a lot of possibilities, even outside the language programming area: for example, questions for HTML pages design or any technology with a syntax or content analysis tool.

4 Methodology

The self-assessment module can be used for learning-teaching of programming languages in several scenarios. It has been used in the ETSII of the UPM for some programming languages courses with the following methodologies:

4.1 Face to Face Teaching. On-line Self-assessments

The system provides a supporting tool for students' homework in face-to-face courses. It improves the motivation, encourages continuous homework and checks the student's comprehension of the subject with an immediate feedback. The application has been used in this manner in a first programming language course about TurboPascal with some hundreds of students per academic term. Five tutors have been scheduled several exercises as they progress along the syllabus subject. Questions are selected randomly from a database of about 800 questions. It has allowed the teachers to assess the students without correcting thousands of exercises [11].

4.2 Face to Face Practical Teaching. On-line Self-assessment during the Class

The application provides a method to keep students' attention during a practical lecture. The tutor should design some questions for a practical session, then explain the problem in the lecture and clarify the doubts from the students. The tool will check the progress of the students and keep them working, therefore the teacher can focus on individual explanations. Even more, while the correcting program will provide some clues about the students' work, the tutor can explain the doubts more efficiently. There is no need for a huge question database because it is not likely the students are going to cheat, just a few long questions will work. This methodology is being used in a course about C programming language with thirty students, one PC per student and face to face teaching.

4.3 Full On-line Teaching

Tutors of online courses use the self-assessment system to track the students' progression during the term, since there are no face-to-face lectures. This methodology is being used in a course about Java programming language with fifteen students from different European countries during the second term of the academic years 2004-05 and 2005-06.

5 Experimentation and Results

The self-assessment system with code question has been tested with students enrolled in a course named Object Oriented Programming taught in the Computer Science Department of the ETSII-UPM during the first term of the 2005-06 academic course. In this course, self-assessment appears as a key activity to encourage the students to connect the students actively in Object Oriented Programming basics by doing. In this way, the Java code questions are absolutely necessary. More than 150 questions have been generated and stored in the courses database. The tutor set up a new self-assessment test after the face-to-face lesson and, in order to encourage the students, the exercises results make a contribution (30%) to the course grading, so these marks are meant not only to motivate but also to assess. This, also, reduces the anxiety of a final examination since each exercise is less important in determining the final grade of the student. On the other hand, frequent exercises also provide a more valid basis for a grade since one bad day has much less of an effect. Students can take each test more than once to improve their score. They can use any book, bibliographic material or reference to solve the questions.

5.1 Results

Fifteen students and a tutor have participated in this course activity. Students finished 275 tests configured by the tutor. Consequently, tutor avoided having to correct one thousand questions during that term.

5.2 Students Opinion

At the end of the term, students completed an anonymous questionnaire, providing a very interesting information and feedback about the course and the methodology. Ease of use, flexibility and instant feedback were seen as one of the major benefits. According to the questionnaires results, 86% of them enjoyed using AulaWeb and 93% of them thought that AulaWeb was very useful. On the other hand, some students had some connections problems. Overall comments were positive, so much that the majority would be pleased if a similar system were used in other courses.

6 Conclusions

The overwhelming conclusion is that this type of blended methodologies is viewed positively by students and tutors. Students do not have to install locally a programming environment in their home computers for training and practice purposes. Academic staff acceptance is also overwhelmingly positive, showing that not only the system is very easy to manage but also has a very intuitive interface and gives very useful feedback to students. Furthermore, teachers do not have to correct programming exercises and the system makes them easy to motivate, track, assess and grade students. Depending on the course characteristics, tutors can choose a different methodology. Moreover, this kind of web-based application may help to reduce distance barriers not only for local or national students but also for other students from international institutions.

The system will be further developed to include code questions for other programming, script or symbolic languages like MatLab, FORTRAN, Maple and SQL, in order to be used in other courses that need this kind of pedagogical tools.

Acknowledgements:

The authors would like to acknowledge the implementation support of A. Alonso, J. M. Arranz, P. Avendaño, M. Aza, J. A. Criado, F. de Ory, C. Engels, M. Fernández, P. García, M. González, J. Granado, T. Hernández, I. Iglesias, J. A. Jaén, A. R. López, D. López, J. A. Martín, M. Martín, F. Mascato, D. Molina, C. Moreno, L. M. Pabón, J. C. Pérez, A. Rodelgo, A. Valero, E. Villalar and C. Zoido.

References:

[1] García-Beltrán, A., Martínez, R.: Challenges of a blended e-learning system in traditional engineering faculties, *Proc, of 2nd Int. Conf. on Multimedia and Information & Communication Technologies in Education*, Vol. III, Badajoz, Spain, December 3-6th (2003), 1960-1963

[2] Wankat, P.C., Oreovicz, F.S.: *Teaching Engineering*. Purdue University (1990)

[3] Venables A. and Haywood, L.: Programming students NEED instant feedback!, *5th Australasian Computing Education Conf. (ACE2003)*, Adelaide, Australia, Conf. in Research and Practice in Information Technology, Vol. 20. T. Greening and R. Lister, Eds., (2003).

[4] F. Rizvanov and R. Lizotte, A Bridge to Success: Active Learning Model for the Effective Hybrid Courseware Development, *Proc. of Fourth International North America Web Conference*, Fredericton, New Brunswick, (1998) 181-184.

[5] N. Serbedzija, A. Kaiser and I. Hawryszkiewycz, E-Quest: A Simple Solution for e-Questionnaires, *Proc. of the IADIS International Conference e-Society 2004*, Avila, Spain, (2004) 425-432.

[6] D. Smith and G. Hardaker, e-Learning Innovation through the Implementation of an Inter-net Supported Learning Environment, *Educational Technology & Society*, 3(3), (2000), 422-431

[7] S. Lewis and G. Mulley, Experiences gained from producing a compiler to guide first year programming students, *5th Annual Conf. on Teaching of Computing*, Dublin, (1997) 129-131

[8] E. V. Wilson, ExamNet asynchronous learning network: augmenting face-to-face courses with student-developed exam questions, *Computers and Education*, 42, (2004) 87-107.

[9] M. Thelwall, Computer-based assessment: a versatile educational tool, *Computers and Education*, 34, (2000) 37-49.

[10] R. Martínez, A. García-Beltrán. AulaWeb: a WWW-Based Course-Support System with Self-Assessment and Student Tracking, *Proc. of ED-MEDIA 2001, World Conf. on Educational Multimedia, Hypermedia and Telecommunications*, Tampere, Finland (2001) 1239-1240.

[11] García-Beltrán, A., Martínez, R.: Web assisted assessment in computer programming learning using AulaWeb, *International. Journal of Engineering Education*, 22-5 (2006)

[12] González, M. *Sistema de Autoevaluación con Preguntas de Programación Multilenguaje Integrado en la Plataforma de eLearning AulaWeb*, Ms. Thesis, Universidad Politécnica de Madrid, Spain (2005).

[13] GNU Operating system web page. Retrieved from the World Wide Web at http://www.gnu.org/software/make/ on April 26th 2006.