Int. J. Engng Ed. Vol. 25, No. 2, pp. 221–227, 2009
Printed in Great Britain.

0949-149X/91 $3.00+0.00

Simulator for a Multi-Programming Environment for Computer Science Learning and Teaching*

A. GARCÍA-BELTRÁN, S. TAPIA, R. MARTÍNEZ and J. A. JAÉN
Dpto. de Automática, Ingeniería Electrónica e Informática Industrial—Universidad Politécnica de Madrid C/ José Gutiérrez Abascal 2, 28006-Madrid, Spain. E mail: agarcia@etsii.upm.es

The objective of this paper is to explore improvements in the learning process for Computer Science using a new tool (an IDE simulator) and to demonstrate the pedagogic and cognitive aspects of the tool. This work presents the design and implementation of a web-based self-assessment environment with multi-language programming questions. The application has been implemented in a complete e-learning system, known as AulaWeb, and is being used as a facility to encourage students on computer science courses to practice programming techniques with different programming languages, for example, Java and C/C++. Furthermore, this paper describes the pedagogical methodology and some results drawn from the experience.

Keywords: IDE simulator; web-based systems; programming languages; learning-teaching strategies; assessment

INTRODUCTION

PROGRAMMING is an essential topic for first courses of an engineering curriculum. Basic programming techniques may be required in later courses and in the professional career to follow. New technologies and tools can help and effectively change the classical approach to learning/teaching, taking lessons, assignments and manual grading [1] with many educational benefits and at a low cost [2]. In this way, web systems that support the marking and grading of students' answers for programming assignments may be efficiently used in the pedagogical process: students have spatial and temporal flexibility to do the assignments and also immediate feedback, and teachers can support large groups of students, avoid grading discrepancies [3] and focusing their activities on issues of content and didactics [4–7].

Nowadays, there is a variety of web systems for the assessment of programming assignments [8]: some of them are commercial and have a 'per-seat' cost for students [9], other systems are not incorporated into a learning management system (LMS) that facilitates e-learning functions such as the management of the students and courses [10], and most of them are oriented only to some specific programming language [11–15].

In this work, we are going to present a web-based simulator for a multi-programming IDE (Integrated Development Environment). This simulator has a unique set of characteristics: it is non-commercial, LMS-embedded, multi-programming language support and is easy to learn and use and can be exploited as an assessment system to generate, deliver and (automatically) grade programming assignments. In fact, we have taken advantage of it as a resource to assist engineering students to learn and practice computer programming techniques, and we will describe how it can change and improve the learning process in the following aspects.

1. Competence training. The simulator is focused on running programs not on learning language syntax or program rules. Of course the learning of the language syntax and rules are a previous step.
2. Student motivation. The proposed problems can be sorted in order of increasing difficulty.
3. Continuous learning. The exercises can be scheduled using deadlines, so the students must work according to the subject design.

LMS SYSTEM AND TEACHING APPROACH

The simulator is embedded, by means of a Java applet, in a self-assessment module included in a web-based LMS (Learning Management System) known as AulaWeb [16]. Students need only a computer connected to the Internet and a web browser in order to take advantage of the simulator. This LMS was developed by the Computer Science Department of the Escuela Técnica Superior de Ingenieros Industriales of the Universidad Politécnica de Madrid (ETSII-UPM) and has been used as an on-line support for courses by more

* Accepted 2 November 2008.

than ten thousand UPM students since 1999. This LMS exploitation consists of four key activities in more than seven hundred courses: theoretical and practical content, open discussion forums, self-assessment exercises and homework delivery. Each tutor can make use of these activities depending on the course methodology and its characteristics, i.e. self-assessment may be seen as a major activity in programming courses with a massive number of students. The main challenge for tutors is in encouraging the students to engage actively in learning programming fundamentals, and, in this way, a regular assessment system is essential.

Up to 2001, there were several ordinary types of questions (single-choice, multiple-choice, short answer, true–false, etc.) implemented on the Aula-Web platform. However, in a programming course, tests should be driven preferentially to questions with code answers. From the 2001/02 academic year onwards, the environment had specifically been made suitable for self-assessing computer programming skills in TurboPascal language [17]. In this case the system asks the student a question with a TurboPascal code-type answer in order to carry out a specific task and subsequently correct his/her answer automatically and immediately. Tutors can create the database of questions, with a friendly and easy-to-use interface for adding and updating questions. They can also configure exercises indicating the quantity, level of difficulty, type and syllabus chapters of the questions. When the exercise is finished, the results are stored in the database, and the system allows the student to check the exercise, 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 system allows the teacher to track the student's progress during the course and also provides some statistical tools for comparing the theoretical and experimental level of difficulty of the questions, revising the initial questions. The feedback from this use has resulted in improved systems and methodologies incorporating great experience and best practices. The success of this system has encouraged the system developers since 2005 to take into account other (any) programming languages such as C/C++ or Java for the exercise questions.

IDE SIMULATOR DESCRIPTION

The multi-programming IDE simulator interface integrates a Borland-type text editor, a set of on-line compilers and an automatic test generator for logical checking and debugging (Fig. 1).

The development of each programming code question by the teachers involves the following components: 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. The wording determines the assignment that the code must carry out. For example: *Complete the following Java application in order to work with the Rectangle class in the RectangleTest program . . .*

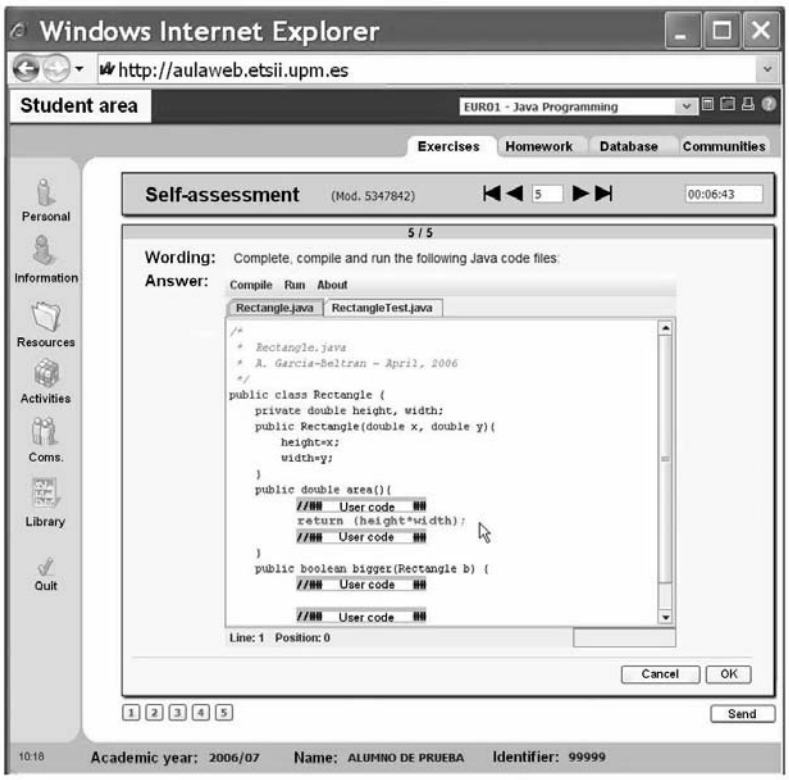

Fig. 1.

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 filled in by the students. In the simplest question, 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 in and clicked over the Compile option, the answers are sent to the server and the complete code files are compiled. Subsequently, the server gives back the result of the compilation. Finally, the code files for the correction 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 [18]. Once the student has filled them in and clicked 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 system can deal with multi-programming language, since the compiling and running process for testing the answers is fully configurable by means of a *makefile*. The interface is the same, but the question design changes. 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 from a wide variety of tools or even design his or her own tools. So far, questions for Java and C/C++ programming languages have been designed and implemented.

Students can use the windows of the text editor to input and edit the answer code and the menu bar options to compile and execute the code. The Compile option compiles the code in the edit window. Different on-line compilers have been installed on the server in order to support this automatic checking of the code answers for the corresponding languages. When compiling is complete, a status window appears. As in a real programming environment, this window indicates an error message if a compile-time or syntax error occurs. If the student answers have no compilation errors, then the event window indicates a message of *Successful Compilation*. The Run option executes the input code, together with a checking code in order to detect run-time or logical errors. In both cases, the corresponding error windows are shown. For self-assessment in logical errors, student code outputs are compared to some model outputs (sometimes randomly generated). Each specific error may require the display of a customized message previously written by the author of the question. If the student answers do not generate any logical or run-time errors, then the window indicates a message of *Successful Execution*.

LEARNING-TEACHING ASPECTS

The multi-programming IDE simulator can be used for the learning–teaching of programming languages in several situations. The following scenarios belong to some programming courses taught in the ETSII-UPM.

Traditional teaching with on-line self-assessments

The system provides a support 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 immediate feedback.

Practical teaching with on-line self-assessment in the computer laboratory

The application provides a method to keep the 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 that the students may have. 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 that the students are going to cheat, just a few long questions will work.

On-line teaching with self-assessment

Tutors of online courses use the self-assessment system to track the students' progress during the academic term, since there are no face-to-face lectures.

EMPLOYMENT AND EXPERIENCES

The educational staff of the ETSII-UPM Computer Science Department has taken advantage of the simulator in the following courses for several degree programs (each credit corresponds approximately to ten hours of traditional face-to-face lectures) in the three different pedagogic case studies since 2005.

Case study (1) 3808—Object Oriented Programming course, 6 credits (traditional lectures, online self-assessment and about 15 students per course). The tutor schedules several exercises as the course progresses through the syllabus. Questions are selected randomly from a database of about 200 questions. It has allowed the teachers to assess the students without correcting hundreds of answers. In any case, to motivate the student and encourage the use of the simulator, the results of the corresponding programming test set up by the teachers are saved in the system database and make an extra contribution to the course grading, so these marks are meant more to motivate than 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 use the simulator as many times as they want to improve their score. In addition to this, the system records allow the teacher to track the students' progress during the course. Table 1 summarizes the exercises set up by the tutor, the number of questions that make up an exercise, the number of the students who did each exercise, the total number of exercises done by the students and the corresponding average score in the academic year 2007–08.

Case study (2) 9006—C/C++ Programming course, 3 credits (lectures at the computer laboratory and about 30 students per course). The simulator is used in this course about C programming in the computer laboratory with one PC per student and face to face teaching. A set of C/C++ code questions has been developed for this course and self-assessment exercises are done during the class. The pedagogical target of the code questions is oriented not only to C/C++ syntax but also to algorithms development. Table 2 shows the results of this activity in 2007–08.

Case study (3) 9122—Java Programming course, 4.5 credits (online, about 30 students per course). In this case, the system is used in a Java programming course with students from different European countries. There are no face-to-face lectures and self-assessment appears as a key activity to encourage the students to connect actively in Java basics by 'doing'. In this way, the Java code questions are absolutely necessary. More than 150 questions have been generated and stored in the course's database. The tutor set up a new self-assessment test after the online lesson and, in order to encourage the students, the exercise results contribute (30%) to the course grading, so these marks are meant not only to motivate but also to assess. Students have only one attempt per exercise, so they cannot improve their score by repeating the exercise. They can use any book, bibliographic material or reference to solve the questions. Table 3 presents the results of this activity in 2007–08.

ASSESSMENT OF THE SIMULATOR

To check the effectiveness of the simulator, students completed questionnaires at the end of the academic period, providing anonymous and very interesting information and feedback about the simulator. First, they were asked to answer a set of questions. The responses were given a five-position scale graded from 1 (Strongly disagree) to 5 (Strongly agree). Results are shown in Table 4 (Object Oriented Programming), Table 5 (C/C++ Programming) and Table 6 (Java Programming).

The results of the questionnaire for the students of all the different courses were conclusive: the use of the IDE simulator was very positive in all the case studies. The best results can be found in the

Table 1. Summary of exercise results in OOP course (2007–08).

	Unit	From:	To (deadline):	Number of questions	Students who did it	Total done	Av. Score (out of 10)
1	Introduction	12/10/2007	25/10/2007	5	10	27	7.41
2	Java Basics	25/10/2007	07/11/2007	5	10	23	7.83
3	Objects and classes	08/11/2007	20/11/2007	5	11	23	6.96
4	Inheritance	22/11/2007	10/12/2007	4	9	12	7.29
5	Interface and packages	05/12/2007	14/12/2007	4	6	13	7.5
6	Streams	15/12/2008	10/01/2008	4	12	36	7.99
7	Threads	17/01/2008	24/01/2008	4	9	21	8.69
8	GUIs	20/01/2008	29/01/2008	4	9	16	8.44

Table 2. Summary of exercises results in C/C++ programming course (2007–08).

	Unit	From:	To:	Number of questions	Students who did it	Total done	Av. Score (out of 10)
1	Introduction	18/02/2008	03/03/2008	1	27	27	10
2	Variables and types	07/03/2008	14/03/2008	2	25	25	9.2
3	Pointers 1	22/03/2008	29/03/2008	1	22	22	9.58
4	Pointers 2	27/03/2008	04/04/2008	5	23	23	9.92
5	Arrays 1	04/04/2008	11/04/2008	1	23	24	10
6	Arrays 2	11/04/2008	18/04/2008	3	22	22	10
7	Cash-machine simulator	18/04/2008	25/04/2008	1	22	25	8.8
8	Bubble algorithm	29/04/2008	06/05/2008	1	23	23	10
9	Find roots of second-degree polynomial	02/05/2008	09/05/2008	1	23	23	10
10	Files. Reading XML	05/05/2008	12/05/2008	3	10	26	6.67
11	Libraries	09/05/2008	16/05/2008	1	8	8	10

Table 3. Summary of exercise results in Java Programming course (2007–08).

	Unit	From:	To (deadline):	Number of questions	Students who did it	Total done	Av. Score (out of 10)
1	Introduction	03/03/2008	10/03/2008	5	21	21	9.05
2	Program structure and data types	07/03/2008	14/03/2008	5	19	19	9.05
3	Operators	22/03/2008	29/03/2008	5	24	24	9.17
4	Control statements	27/03/2008	04/04/2008	4	24	24	9.25
5	The return statement	04/04/2008	11/04/2008	4	25	25	9
6	Objects and classes	11/04/2008	18/04/2008	4	26	26	8.46
7	Class members	18/04/2008	25/04/2008	4	25	25	8.6
8	Inheritance	29/04/2008	06/05/2008	5	27	27	9.26
9	Interfaces	02/05/2008	09/05/2008	4	27	27	9.07
10	Packages and exceptions	09/05/2008	16/05/2008	4	24	24	9.06

Table 4. Summary of the assessment of the simulator in an Object Oriented Programming course (2007–08).

	Question	Students	Answers	1	2	3	4	5	Av.
1	I enjoyed using web assisted assessment.	10	9	0	0	0	2	7	4.78
2	The simulator enabled me to practice and develop programming skills.	10	10	0	0	0	6	4	4.4
3	I worked harder than I would have done without it.	10	8	0	0	0	2	6	4.75
4	It encouraged me to work consistently throughout the term.	10	10	0	0	0	1	9	4.9
5	I would like the end of semester examination to be taken using the simulator.	10	8	0	0	0	1	7	4.88
6	The results have been a fair reflection of my ability.	10	8	0	0	0	3	5	4.62

Table 5. Summary of the assessment of the simulator in a C/C++ programming course (2007–08).

	Question	Students	Answers	1	2	3	4	5	Av.
1	I enjoyed using web assisted assessment.	22	22	0	0	4	9	9	4.23
2	The simulator enabled me to practice and develop programming skills.	22	22	0	5	6	8	3	3.41
3	I worked harder than I would have done without it.	22	22	0	0	6	11	5	3.95
4	It encouraged me to work consistently throughout the term.	22	22	0	1	3	9	9	4.18
5	I would like the end of semester examination to be taken using the simulator.	22	22	0	5	4	7	6	3.64
6	The results have been a fair reflection of my ability.	22	22	0	0	4	12	6	4.09

Table 6. Summary of the assessment of the simulator in a Java Programming course (2007–08).

	Question	Students	Answers	1	2	3	4	5	Av.
1	I enjoyed using web assisted assessment.	23	21	2	1	3	5	10	3.95
2	The simulator enabled me to practice and develop programming skills.	23	20	0	1	4	5	10	4.2
3	I worked harder than I would have done without it.	23	21	2	1	4	5	9	3.86
4	It encouraged me to work consistently throughout the term.	23	21	1	2	4	6	8	3.86
5	I would like the end of semester examination to be taken using the simulator.	23	19	2	1	3	3	10	3.95
6	The results have been a fair reflection of my ability.	23	21	0	1	4	7	9	4.14

traditional course (Object Oriented Programming course), but the results in the others were also very satisfactory.

Moreover, on this questionnaire students also had the chance to put forward written comments in order to explain or develop their scored responses. The written comments are closely correlated with previous responses. Flexibility, ease of use and instant feedback were seen as some of the major benefits. On the other hand, many students had problems with their Internet connection, particularly using certain Internet providers due to their proxy systems. In any case, overall comments were very positive: teachers do not have to correct programming exercises, and students do not have to install programming environments locally in their home computers and to be in contact with their teachers for training and practice purposes.

CONCLUSIONS

The use of this kind of web-based IDE simulators is viewed positively by students and tutors. Students do not have to install a programming environment locally on their home computers for training and practice purposes. Academic staff acceptance is also overwhelmingly positive, showing that the system not only 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 it easy to motivate, track, assess and grade students. Depending on the course characteristics, tutors can choose a different methodology approach. 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.

For all these reasons, it is worth developing this kind of simulator for at least one year, including application design, software implementation, content development and validation phases. Although in this type of project, tutors and software developers must keep on working indefinitely to support online content and activities and to update technologies.

Acknowledgements—This work is partially funded by the División de Informática Industrial of the Universidad Politécnica de Madrid. 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, 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. P. C. Wankat and F. S. Oreovicz, *Teaching Engineering*, Purdue University, (1990).
2. P. Ball and H. Thornbury, A student learning environment without the overhead? Reviewing cost and benefits of CAL within a manufacturing course, *International Journal of Engineering Education*, **20**(5), 2004, pp. 713–725.
3. J. M. Montoro, R. San-Segundo, J. Macías-Guarasa, R. de Córdoba and J. Ferreiros, Methodology for the analysis of instructors' grading discrepancies in a laboratory course, *International Journal of Engineering Education*, **22**(5), 2006, pp. 1053–1062.
4. M. Amelung, P. Forbrig and D. Rösner, Towards generic and flexible web services for e-assessment, *Proc. ITICSE'08*, Madrid, Spain, (2008) pp. 219–223.
5. M. S. Perez, P. Herrero, F. M. Sanchez, V. Robles, Are web self-assessment tools useful for training?, *IEEE Transactions on Education*, **48**(4), 2005, pp. 457–463.
6. S. Hussmann, G. Covic and N. Patel, Effective teaching and learning in engineering education using a novel web-based tutorial and assessment tool for advanced electronics, *International Journal of Engineering Education*, **20**(2), 2004, pp. 161–169.
7. M. J. Gallego and V. Gámiz, La plataforma AulaWeb en la enseñanza práctica de los estudiantes de Educación, *eUniversaLearning: Congreso Internacional de Tecnología, Formación y Comunicación (eUniversaLearning 07)*, Salamanca, Spain, (2007).
8. P. Brusilovsky and C. Higgins, Preface to the special issue on automated assessment of programming assignments, *Journal of Educational Resources in Computing*, **5**(3), 2005.
9. O. Gotel, C. Scharff and A. Wildenberg, *Extending and Contributing to an Open-source Web-based assessment system for the automated assessment of programming problems, Proc. ACM Conference on Principles and Practices of Programming in Java*, Lisbon, Portugal, (2007).
10. S. H. Edwards, Improving student performance by evaluating how well student test their own programs, *J. Educational Resources in Computing*, **3**(3), 2003, pp. 1–24.
11. S. P. Foubister, G. J. Michaelson and N. Tomes, Automatic assessment of elementary Standards ML programs using Ceilidh, *Journal of Computer Assisted Learning*, **13**, 1997, pp. 99–108.
12. A. C. Croft, M. Danson, B. R. Dawson and J. P. Ward, Experiences of using computer assisted assessment in engineering mathematics, *Computers & Education,* **37**, 2001, pp. 53–66.
13. N. Catenacci and L. Sommaruga, The evaluation of the Hyper Apuntes interactive learning environment, *Computers & Education* **32**, 1999, pp. 35–49.
14. 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, 4, (2004) pp. 25–432.
15. M. Thelwall, Computer-based assessment: a versatile educational tool, *Computers and Education*, **34**, 2000, pp. 37–49.
16. A. García-Beltrán, R. Martínez, AulaWeb: un sistema para la gestión, evaluación y seguimiento de asignaturas, Industria XXI, **2**, (2001) 11–16.
17. A. García-Beltrán, R. Martínez. Web assisted assessment in computer programming learning using AulaWeb, *International Journal of Engineering Education*, **22**(5), 2006, pp. 1063–1069.
18. R. S. Pressman, *Software Engineer. A Practitioner's Approach*. McGraw-Hill, (2002).

Web Address. More information is available via the Internet at URL: http://www.dii.etsii.upm.es/aulaweb

Ángel García-Beltrán is an Assistant Professor in the Dpto. de Automática, Ingeniería Electrónica e Informática Industrial of the Universidad Politécnica de Madrid. He teaches several Java Programming and Object Oriented Programming courses and is also active in innovative engineering education methods using the web.

Santiago Tapia is an Assistant Professor in the Dpto. de Automática, Ingeniería Electrónica e Informática Industrial of the Universidad Politécnica de Madrid. He teaches C/C++ Programming courses and is active in engineering education using the Internet.

Raquel Martínez is an Assistant Professor in the Dpto. de Automática, Ingeniería Electrónica e Informática Industrial of the Universidad Politécnica de Madrid. She has been teaching since 1979 in a variety of Computer Science topics and is actively involved in web-based educational projects.

José-Alberto Jaén is a Full Professor in the Dpto. de Automática, Ingeniería Electrónica e Informática Industrial of the Universidad Politécnica de Madrid. For the past 30 years he has been teaching Mathematics and Computer Science courses.