Int. J. Engng Ed. Vol. 22, No. 5, pp. 1063–1069, 2006
Printed in Great Britain.

0949-149X/91 $3.00+0.00

Web Assisted Self-assessment in Computer Programming Learning Using AulaWeb*

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

This work discusses some issues associated with Web assisted self-assessment of the computer programming skills of first-year undergraduate engineering students. The self-assessment exercises are meant to encourage and motivate them rather than to assess them. The whole 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 issues of constructing suitable tests and their set up, implementation and results are described. Finally, the results of a survey of students' perceptions and the influence of these on future developments are presented.

Keywords: teaching/learning strategies; programming and programming languages; evaluation methodologies; interactive learning environments.

INTRODUCTION

THE MAIN PURPOSE of this paper is to describe the implementation of a Web-based self-assessment environment. This application has been implemented in a complete e-learning system, named AulaWeb [1], and has been used as a facility to encourage first-year (and first-semester) students of engineering to practice computer programming techniques. Furthermore, this work presents the pedagogical methodology and the results drawn from this experience. The AulaWeb system has been used as an on-line support for courses by more than three thousand students at the Escuela Técnica Superior de Ingenieros Industriales of the Universidad Politécnica of Madrid for the last five years. From the 2001/02 academic year onwards, the environment has been specifically made suitable for self-assessing computer programming skills. The feedback from this use has resulted in improved systems and methodologies incorporating new implementations, great experience and best practices.

Computer programming might only make up one twelfth of an engineer's first year program. At first-year level it is important that students develop confidence and competence in the basic programming techniques that will be required later in their engineering studies and/or professional tasks. A major challenge for teachers is to encourage the majority of students, whose primary interest is not computer programming, to engage actively in learning programming basics. To be successful, this engagement must begin sufficiently early in the course and there must be enough motivation to continue to practice until the end of the semester. In this way, self-assessment is fundamental in training and education [2–4].

Computer-Based Assessment (CBA) or, more particularly, Web Assisted Assessment (WAA) has made this possible through a regular testing system with large groups of students and few lecturers. Many authors report on different WAA systems [5–9] and some universities have implemented some kind of assessment tools [10–14]: it would be impractical, tedious and slow to implement this kind of assessment without a computer. The use of a computer also ensures consistency and accuracy in the correction process. In addition to self-assessment, Web systems can support other formative functionalities. In this case AulaWeb system also supports the provision of news, work collecting, lecture notes, Web pages, links, questionnaires and chats. However, the current work focuses only on how 695 first-year engineering students on a traditional programming course were tested at least eleven times during the semester using a WAA system with not only formative but also summative aims [15].

PRACTICE ENCOURAGING WITH WEB ASSISTED ASSESSMENT

At first year level, assessment strategies are generally based upon verifying that a sufficient number of computer programming skills have been learned. A traditional written examination is held at the end of the semester. Because of the time required to practice and develop programming skills, those students who do not start to engage in the learning process sufficiently early find themselves doing badly in examinations and obtaining low marks. In this experience there are eleven WAA tests during the semester. These are

* Accepted 18th January 2006.

used to improve students' performance, focus the students' activities and so drive then to practice computer programming during the academic period. To encourage students, the test questions are similar (in form and level of difficulty) to the final written examination and the test results make a contribution to the subject grading. Students will more easily accept a regular test system if they know that the results will have some influence on their final marks. In this case, the WAA element can increase the final subject mark by one additional point (out of ten points). Students can take each test more than once (with other randomly selected questions from the database following the template set up by the tutors and always before the deadline) to improve their score. The mark contribution of each test is the average mark of all these grading test attempts. Anyway, traditional written exams are still used for most of the final grading so these exercises are meant to motivate rather than to assess. The system aims to get students to engage positively in this formative approach: the students are also allowed to try customized and randomly selected tests with the large database questions (identical to the real tests) as many times as they want during the semester. In these trial tests students gain familiarity with the subject area, the system environment and the type and level of difficulty of the questions. The strategy that students usually employ is that they do these tests several times until their knowledge level is sufficient to proceed to the actual grading test. This pedagogical model, including a large number of questions, randomly selected tests, instant feedback and computer programming assessment, would be impossible to implement with a traditional written assessment system and such a large number of students. Moreover there are other benefits: reducing the risk of cheating, results management facilities, tests can be used every year and students can take their tests at different times and in different places (spatial and temporal flexibility), etc.

DESCRIPTION OF THE ASSESSMENT PROCEDURE

Students must be able to log into the AulaWeb e-learning system using their username and password. No serious problems with access have been reported except the day before the deadline of some tests because of a little collapse of the Web server. All the tests were available anywhere (on campus or at home) by WWW but the large number of questions in the database limits the opportunity for plagiarism. Anyway, the small contribution of the test results to the final mark (10%) reduces this issue significantly. Student behavior can be tracked to include details of the number of attempts made at the various tests, the time and date the tests are accessed and the time spent taking each test.

The student can navigate through the self-assessment environment (Fig. 1) using a set of buttons (bottom left-hand corner) and/or controls (top center). At any stage students may amend an answer they have typed in or save the session and postpone the end of the test. Once the student has finished the test, the answers are submitted to the AulaWeb server for marking: the feedback for students and teachers is automatic and immediate. However, in both types of test, real and trial, feedback is given in the form of a numerical mark out of ten points at the end of the test.

Within this environment the authors of this paper have inputted a large number (823) of questions into the database server since 1999. The users (five tutors and hundreds of students) have therefore had sufficient time and opportunity to obtain comprehensive feedback and perform an exhaustive quality control: discovering any mistake in the wordings or in the answers and correcting them if necessary.

The questions have been categorized into eleven topics or units and five levels of difficulty and they can incorporate graphics and multimedia. There are also many question types (single choice, multiple choice, numeric input, text, etc.) implemented in this tool but, in this course, use has been preferentially given to questions with TurboPascal code answers and questions with randomly generated values (parameters). These question types are more suitable to the programming skills of the users (i.e. 'Complete the following subroutine in order to return the average value of the numbers in a file . . .') and the knowledge ('Convert the following binary representation to its equivalent decimal form: 011010—or the randomly generated value') to be discussed in this Computer Science course. Not only the questions, but also the course syllabus and its content are progressive and cumulative, although in some topics it can test stand-alone concepts.

Questions with TurboPascal code answers

A piece of code or program is correct if it can be proved to meet its specification. Correctness assessment is empirical and based on checking a code's output from inputs. In CBA, student code outputs are compared automatically with model outputs [2].

Figure 1 shows a typical question interface that includes a *virtual* Borland-type programming environment, implemented by means of a Java applet. However, questions are designed to be not too long as they should be displayed in one full screen. Students can use the edit window of the virtual environment to input and edit the answer code and the menu bar options to compile or execute the code. The *Compiler* option compiles the code in the edit windows. An on-line compiler has been installed in the server in order to support this automatic checking for the Pascal code answers. When compiling is complete, a status window appears. If a compile-time or syntax

Fig. 1. A test example with a TurboPascal code question in the student interface.

error occurs, an error message is shown like in a real programming environment. The *Ejecutar* 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. Each specific mistake may require the display of a customized error message previously written by the lecturers. As in other types of questions, when a student is satisfied with his or her answer, they can save it with the assessment session. However, teachers do not have to correct programming exercises and students do not have to install a Pascal programming environment locally in their home computers for training and practice purposes. Other types of questions have been used mostly for the first unit (*Introduction*) of the subject because of its theoretical and fundamental (no programming) contents.

Questions with randomly generated values

This type of question allows the generation of an effectively infinite set of questions. This is a very positive circumstance not only for educational reasons (students can practice as much as they like) but also for security reasons (reducing risk of cheating). The drawbacks are that time is needed to prepare the question and a program language knowledge is needed to write the code to develop the question wording with random factors. In this case, the program language is also TurboPascal.

TEST SET UP

During the term, teachers inform the students about the programming of a new exercise template that is associated with and according to the end of the unit taught in the face-to-face sessions:

1. Each exercise has a deadline of between two and four weeks from that date, in order to leave the students enough time to complete the exercise but to encourage them not to leave the work until the end of the semester. Although the lecturer can set a time limit for each assessment session there is none (except the deadline for each exercise).

2. Since the rhythm of the subject program teaching in the face-to-face sessions in the traditional classroom are similar for all the lectures, the exercise deadline for the corresponding groups of students differs only by one or two days. The aim is not to jam up the AulaWeb server.
3. Students can do the exercises using any computer (at home or in the campus) and a standard Web browser with a Java virtual machine. The aim is to get the students to program regularly, so we do not mind if the students work together on the trial tests or use a real TurboPascal programming environment to work through some of the questions. Following this procedure, the higher-level skills are assessed at the end of the semester.
4. Each exercise contains ten randomly selected questions from the corresponding unit database (except exercise #9, with only six questions) following the template set up by the teachers. Each question is taken randomly from a database of up to 100 questions per unit so that the likelihood of any two tests being the same is very small. Because of the subject program contents, questions with TurboPascal code answers appear in exercises from unit #5 to unit #11.
5. Once the exercise has been generated, the student can browse over the set of questions that make up the test. Students can print out the text of the exercise questions before or after answering them and take the exercise home to think over the answers, change them, etc. before finishing the test. They can modify the input answers and interrupt the exercise session at any time, saving the test and the answers. Later they can carry on with the exercise in another session.
6. Results are shown as an instant feedback once the exercise has been finished by the student. These results are saved in the student database and can be checked on-line at any time by the student. The corresponding lecturer can also consult these data to track the progress of an individual student. Both can print out the feedback and take it home to analyze the mistakes.
7. If the student is not satisfied with their exercise mark, he/she can repeat the corresponding exercise (with other randomly selected questions) as many times as he or she wants. In this case, the final exercise grade is the average of all the completed attempts made by the student.
8. Furthermore, students can, beforehand and without any influence on the grading, do all the exercises they want, in order to familiarize themselves with the system and the difficulty level of the questions. This procedure can also be useful 'to critique and revise the database questions that subsequently were drawn up for course exercises' [14].

TEST RESULTS

In the academic year 2003/2004, nearly 500 engineering students were regularly tested using the system. Figure 2 shows the date distribution for the real tests taken during the first semester of 2003/2004. Note that a large number of students took the tests close to the corresponding deadline.

While students are given instant and individualized feedback on the corresponding tests, lecturers are provided with the following reports:

1. Global statistics showing the total number of students, total number of finished exercises, number of questions (total, right, wrong and

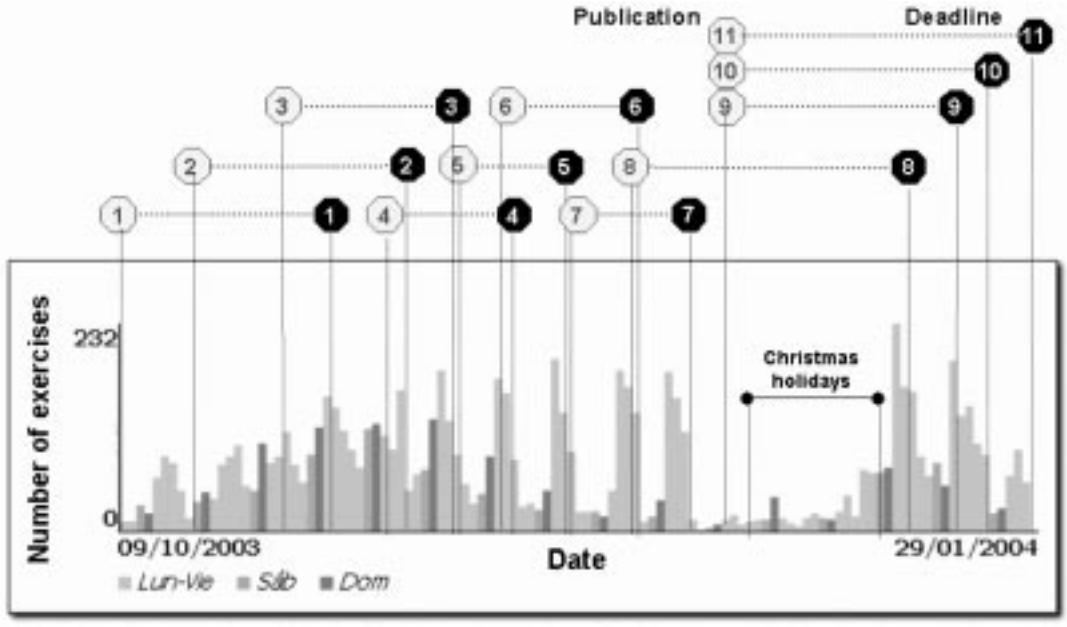

Fig. 2. Date distribution for the exercises done by the students. Publication: date of the exercise template set up. Deadline: date until the exercise may be done.

Ejercicios programados realizados

Nº	Código	Tipo	Variante	Fecha	Tiempo	Nº Preg.	Bien	Mal	S/C	NOTA	
1	IFQ001	No resta	73859804	21/10/2003	4	10	10	0	0	10	
2	IFQ002	No resta	27100771	22/10/2003	9	10	9	1	0	9	
3	IFQ003	No resta	77242678	04/11/2003	9	10	9	1	0	9	
4	IFQ004	No resta	33671206	19/11/2003	52	10	6	4	0	6	
5	IFQ005	No resta	25703066	28/11/2003	29	10	10	0	0	10	
6	IFQ006	No resta	87313478	09/12/2003	38	10	8	2	0	8	
7	IFQ007	No resta	59589785	15/12/2003	39	10	9	1	0	9	
8	IFQ008	No resta	33112734	09/01/2004	48	10	10	0	0	10	
9	IFQ009	No resta	4346865	09/01/2004	2	5	5	0	0	10	
10	IFQ010	No resta	43293398	13/01/2004	29	10	9	1	0	9	
11	IFQ011	No resta	18606204	22/01/2004	21	10	8	1	1	8	
Nota media (sobre 11 ejercicios realizados)										8,91	

Fig 3. An example of an exercises results report of a student.

not answered), score histogram . . . In this course, 96% of students finished, at least one exercise and 8.846 exercises (84.155 questions) were done by all the students.

2. The list of exercises including identifier, title (corresponding unit), number of questions, publication date, deadline, number of students, number of attempts, average number of attempts per student, and average mark. With these data tutors can calculate the falling rate in the self-assessment system: 90% of students finished the first exercise and 64% of them finished the last exercise.
3. A list of individual attempts at each exercise by a particular student with details of test identifier, date and time, time to complete the test, mark and link to questions and answers (Fig. 3).
4. Performance of each question delivered in terms of the number of appearances and level of difficulty. In this course some programming skills seem to be more difficult for students, for instance, recursive subroutines and operations with data structures (files and lists).

EVALUATION

At the end of the semester, students were given a Web-based questionnaire encouraging them to comment voluntarily and anonymously on their experience of WAA. First they were asked to answer nine questions with responses on a five-position scale, graded from 1 (Strongly disagree) to 5 (Strongly agree). A total of 139 replies were received. The responses are summarized in Table 1.

Students were also asked about the usefulness and other important aspects of AulaWeb. They gave high ratings to easiness (4.59) and usefulness (4.37) of the system. It is clear that the self-assessment system has helped students to assimilate the course topics gradually and has also encouraged them to work harder. The overwhelming conclusion from this feedback is that this type of testing is generally viewed positively by students.

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. In particular, ease of use, flexibility and instant feedback were seen as one of its major benefits. Typical comments were:

> Flexibility to take the test when and where you like.
>
> Regular assessment has encouraged me to practice throughout the term.
>
> I would hardly work this subject without AulaWeb assessment.

Table 1. Summary of the questionnaire answers

Question	Total	1	2	3	4	5	Average
I had already used a Web browser before the start of the course.	139	8	7	7	18	99	4.39
I had used a programming language before the start of the course.	139	89	19	7	12	12	1.83
The system is easy to learn and use.	139	0	1	3	46	89	4.59
The system organization is clear.	139	0	2	24	61	52	4.16
Self-assessment is very useful.	139	0	4	13	48	74	4.37
WAA has encouraged me to work throughout the term.	139	3	7	19	57	53	4.06
WAA results should have more weight in the final mark.	139	3	7	21	21	87	4.28
Other courses should use this WAA.	139	0	8	13	31	87	4.39
I enjoyed using this tool.	139	1	5	11	63	59	4.22

[The system] helps you to understand and study the course.

[The system] lets me test my knowledge level.

[The system] obliges me to be up to date with my course work.

I think that other subjects should have already introduced this system.

Drawbacks: many students had problems with their Internet connection, particularly the day before the tests deadline and also using some internet providers connections with TurboPascal code questions:

Server sometimes collapses.

I cannot find solved exercises in the system.

Download speed of the Webpages and the resources.

I could not answer code questions properly from my home because of my Internet provider.

Overall comments were positive, so much so that the majority would be pleased if a similar system were used in other subjects:

I think that other subjects should implement this regular test system.

Academic staff acceptance has been overwhelmingly positive, showing that not only is the system very easy to manage but it also has a very intuitive interface and gives very useful feedback to the students.

It is very difficult for the tutors to evaluate the improvement of the students' performance in a quantitative way. It is not possible to compare two groups of students under the same academic circumstances but with only one group using the self-assessment (who can discriminate against one group of students?). Tutors can only compare the average marks and other indicators between groups of different academic years (the self-assessment system was not available in 1999–2000). In this way the average mark has been increased from 4.15 (1999–2000) to 4.87 (2003–04) and the students' failure rate has been reduced from 70% (1999–2000) to 55% (2003–04). Furthermore, tutors have verified that the number of students that ask questions about programming skills during the academic period has notoriously increased.

CONCLUSIONS AND FUTURE WORKS

This paper has described the implementation of a self-assessment system for first-year engineering computer programming. The implementation in the AulaWeb system appears to offer some benefits, which are now summarized. This system has been subject to large-scale testing since October 1999 and has enhanced the learning pedagogical process. At least in this course, Web assisted self-assessment is better than traditional written assessment, especially when teachers and students can take advantage of code questions with a certain creative component and with the chance of developing questions with randomly generated wording. Furthermore, with such a large number of students the assessment would be impractical without the system: the automatic assessment reduces the lecturer burden of marking. This advantage saved the time of the academic staff , which can be redirected to other aspects of teaching.

Student feedback has been very positive and student responses indicate that they have worked harder, with more motivation and more consistently than they might have done without such a regular test system. Self-assessment instant feedback helps students to build their confidence in developing programming skills. The strongest complaints were about problems with some Internet traffic jams and some internet providers connections and the questions with code answers.

In the future, AulaWeb system should become a regular assistant of this first-level programming course and in other computer/programming courses at the ETSII-UPM. In the case of questions with code answers the focus must move from correctness to optimality assessment: how good is the code semantic style and how well, accurately and quickly does it do what it is supposed to do?

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, J. Granado, T. Hernández, I. Iglesias, J. A. Jaén, A. R. López, J. A. Martín, F. Mascato, D. Molina, L. M. Pabón, J. C. Pérez, A. Rodelgo, S. Tapia, A. Valero, E. Villalar and C. Zoido.

REFERENCES

1. García-Beltrán and R. Martínez, Challenges of a blended e-learning system in traditional engineering faculties, *Proc. 2nd International Conference on Multimedia and Information & Communication Technologies in Education*, Badajoz, Spain, December 3–6th, Vol. III, (2003) pp. 1960–1963.
2. S. P. Foubister, G. J. Michaelson N. and Tomes, Automatic assessment of elementary Standards ML programs using Ceilidh, *Journal of Computer Assisted Learning*, **13**, 1997, pp. 99–108.
3. A. Venables and L. Haywood, Programming students NEED instant feedback!, *5th Australasian Computing Education Conference (ACE2003)*, Adelaide, Australia, Conferences in Research and Practice in Information Technology, Vol. 20. T. Greening and R. Lister, Eds, (2003).
4. 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.
5. J. C. Burguillo, J. V. Benlloch, J. M. Santos, D. A. Rodríguez and F. Buendía, X-Quest: an open tool to support evaluation in distance learning, *Proc. ED-MEDIA 2001 World Conference on*

Educational Multimedia, Hypermedia & Telecommunications, Tampere (Finlandia), (2001) pp. 220–221.
6. N. Catenacci and L. Sommaruga, The evaluation of the Hyper Apuntes interactive learning environment, *Computers & Education* **32**, 1999, pp. 35–49.
7. E. Foxley, C. Higgins, A. Tsintsifas and P. Symeonidis, The Ceilidh CourseMaster system—An introduction, *4th Java in the Curriculum Conference*, South Bank University UK, (2000).
8. A. Prieto, P. Clemente, J. González, R. Rodríguez and E. Sosa, Student progress evaluation web tool*, Advances in Technology-Based Education: Toward a Knowledge-Based Society, Proc. of 2nd International Conference on Multimedia and Information & Communication Technologies in Education*, Badajoz, Vol. III, (2003) pp. 1819–1822
9. 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) pp. 181–184.
10. 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) pp. 425-432.
11. D. Smith and G. Hardaker, e-learning innovation through the implementation of an Internet supported learning environment, *Educational Technology & Society*, **3**(3), 2000, pp. 422–431.
12. S. Lewis and G. Mulley, Experiences gained from producing a compiler to guide first year programming students, *5th Annual Conference on Teaching of Computing*, Dublin, (1997), pp. 129–131.
13. E. V. Wilson, ExamNet asynchronous learning network: Augmenting face-to-face courses with student-developed exam questions, *Computers and Education*, **42**, 2004, pp. 87–107.
14. M. Thelwall, Computer-based assessment: A versatile educational tool, *Computers and Education*, **34**, 2000, pp. 37–49.
15. R. Martínez and A. García-Beltrán, AulaWeb: a WWW-based course-support system with self-assessment and student tracking, *Proc. of World Conference on Educational Multimedia, Hypermedia and Telecommunications, ED-MEDIA 2001,* Tampere-Finland, (2001).

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

Dr. Ángel García-Beltrán is Assistant Professor, Dpto. de Automática, Ingeniería Electrónica e Informática Industrial, ETS de Ingenieros Industriales, Universidad Politécnica de Madrid.

Dr. Raquel Martínez is Assistant Professor, Dpto. de Automática, Ingeniería Electrónica e Informática Industrial, ETS de Ingenieros Industriales, Universidad Politécnica de Madrid.