International Journal of Engineering Education Vol. 30, No. 6(B), pp. 1741–1748, 2014
Printed in Great Britain

0949-149X/91 $3.00+0.00

Performance and Professional Skills in an Online Java Programming Course for Engineering Students*

ÁNGEL GARCÍA-BELTRÁN, SANTIAGO TAPIA and MARÍA-JESÚS SÁNCHEZ-NARANJO
Universidad Politécnica de Madrid, ETSI Industriales, C/José Gutiérrez Abascal, 2, 28006, Madrid, Spain. E-mail: agarcia@etsii.upm.es; stapia@etsii.upm.es; mjsan@etsii.upm.es

The main purpose of this work is to describe the case of an online Java Programming course for engineering students to learn computer programming and to practice other non-technical abilities: online training, self-assessment, teamwork and use of foreign languages. It is important that students develop confidence and competence in these skills, which will be required later in their professional tasks and/or in other engineering courses (life-long learning). Furthermore, this paper presents the pedagogical methodology, the results drawn from this experience and an objective performance comparison with another conventional (face-to-face) Java course.

Keywords: online course; Java programming; self-assessment; teamwork; proficiency test; life-long learning

1. Introduction

Restructuring of the new curricula and syllabus in the European Higher Education Area (EHEA) is forcing a change of the teaching-learning methodologies in order to focus the promotion of the student independent work and the professional competences acquisition without dismiss to strengthen the student trucking by the academic staff [1–3]. Since the birth of the Web, many educational organizations have developed online versions of academic courses [4–5]. Many authors report on the use of different web-based systems and methodologies to help not only in online courses but also in traditional ones [6–8]. These organizations try to make the most of the web advantages, specially, the spatial and temporal flexibility and the multimedia and interactive contents. Some authors emphasize that online students will improve their lifelong learning skills, because online courses force students to think about their own learning pace and knowledge absorption [9–10]. Other authors criticize that online courses demands digital literacy, complicate other academic activities (i.e. teamwork, students tracking and evaluation. . .) and students need to be able to self-regulate their learning and possibly give themselves a learning goal to achieve [11–13]. Despite the challenges and drawbacks nowadays many universities have some kind of OpenCourseWare (OCW) and/or Massive Online Open Course (MOOC) initiatives [14–16].

The main purpose of this work is to describe the case of an online Java programming course for engineering students to learn computer programming and to practice other non-technical abilities: online training, teamwork skills and use of foreign languages in learning. It is important that students develop confidence and competence in these skills, which will be required later in their professional tasks and/or in other engineering courses (life-long learning). Furthermore, this paper presents the pedagogical methodology, the results drawn from this experience and a performance or success comparison with other traditional (face-to-face) Java course.

Several years ago, the teaching style for engineering courses (including programming course) was fairly traditional, with lectures and laboratory work [17]. Currently, new information and communications technologies can be used as a support to the traditional instructional methods. They can become an interactive learning system helping students to learn the basic concepts of engineering subjects. In computer programming, their interactive nature allows the large group of students not only to study the material and see programming code examples, but also to cooperative work for software development and to edit, compile and run programs, and to evaluate their level of learning.

In this case, a new methodology is used in a Java programming elective and online course with about 30 engineering students per term at the Escuela Técnica Superior de Ingenieros Industriales of the Universidad Politécnica de Madrid (ETSII-UPM) since 2005. There are no face-to-face lectures and online contents, e-contact (email and forums), workgroups and self-assessment appears as key activities to encourage the students to connect actively in Java basics by "doing". In this way, these activities results contribute (100%) to the course grading, so these marks are meant not only to motivate but also to assess. Due to the course origin (a European R+D project) English language was used by teachers and students until 2010 for course contents and communication.

* Accepted 15 July 2014.

At present, AulaWeb is used as a support to the traditional instructional methods [18]. It can become an interactive learning system helping students to learn the basic concepts of computer programming. Its interactive nature allows the large group of students of Computer Science courses not only to study the material and see programming code examples, but also to edit, compile and run programs written in Java, and to evaluate their level of learning. AulaWeb is a WWW-based interactive e-learning system which assists students/teachers to learn/teach courses. The system helps students to learn the course content, to deliver practice exercises and to do self-assessment exercises and, furthermore, provides teachers with the possibility of publishing content, creating and configuring exercises and tracking student learning progress. Students and teachers only need a computer connected to the Internet and a WWW browser in order to take advantage of all the application functions. The system architecture, the graphic user interface design and an on-line help system eases user interaction with the system [19].

2. Teaching-learning methodology

Java Programming is a 4.5-credit online course with about 30 students from different European countries per academic year. The main objective entails that students should develop confidence and competence in basic Java programming techniques. There are no face-to-face lectures neither a traditional final exam so the course emphases are on online contents study, teamwork and innovative problem solving (project-based 'learning by doing' and self-assessment). The implementation of these activities taking into account the three main features in the Bologna Process: (a) anything that involves a student effort should be measured, (b) student feedback should be continuous and (c) the monitoring of the activities should consider the evolution of the course [20–21].

The content of the Java online course is organized as a set of documents, structured into several chapters or modules of the course syllabus in SCORM format [22–23] (Fig. 1). The advantages of these resources are that they provide content at the appropriate level in a well-structured form with consistent nomenclature and include appropriate learning aids such as example problems, objectives, figures, tables, and homework problems at a variety of levels of difficulty. These documents include theoretical explanations, problems, exercises, exams, book references, external link images, diagrams as well as software tools and source programs, which can be edited, compiled and executed by the corresponding programming environment. This course documentation and contents can also be found on the UPM OpenCourseWare project website [24]. In addition a paperback version with the complete Java Programming course contents is available at the ETSII-UPM library [25].

A course open discussion forum is also implemented in AulaWeb. This activity can facilitate the interchange of ideas among teachers and students, who can publish news and express their own ideas, doubts and comments, and ask or answer questions posed by other students or by the tutor. Moreover,

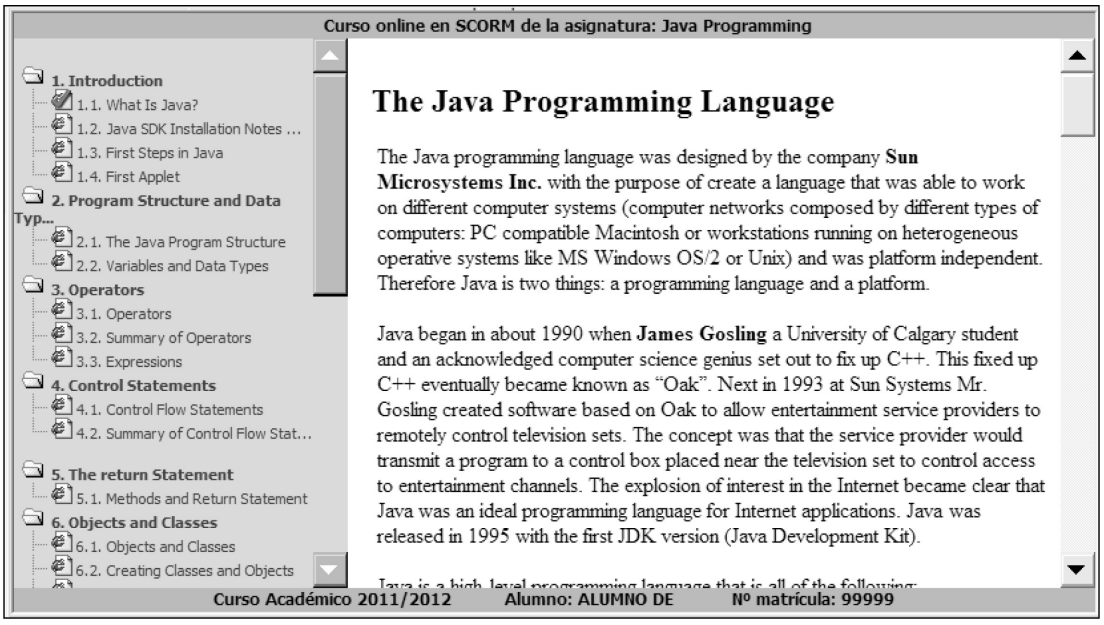

Fig. 1. AulaWeb interface presenting the Java Programming course SCORM content.

some ideas and suggestions, obtained through these forums, were useful to identify topics which required a more detailed and clearer description, or to add new exercises to those courses which were particularly difficult to understand. The student participation in the forum gives the 20% of the course grading. In addition teachers and students can establish complementary communication through the e-mail service.

More than 200 Java code questions have been generated and stored in the courses database. These questions includes a multi-programming IDE simulator interface [26] that integrates a Borland-type text editor, a set of on-line compilers and an automatic test generator for logical checking and debugging (Fig. 2). These questions are used to set up a new self-assessment test after the corresponding chapter. The student can interrupt and postpone the end of the exercise at any time in order to revise the required knowledge. After that, the student can carry on with the exercise. Students have only one attempt per exercise, so they cannot improve their score by repeating the exercise but they can use any book, bibliographic material or reference to solve the questions. When finishing the exercise, the system allows the student the possibility of checking his exercise and comparing his/her answers to the correct solutions. Solution of the test provides the user's level at that moment and updates the values in the database. Evaluation of the exercise is, therefore, automatic, and the student and his/her teacher can access the results of the self-assessment activities. 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. Therefore, the system allows the teacher to track the student's progress during the course since there are no face-to-face lectures.

A final mini-project teamwork is set up to be delivered at the end of the academic period. The AulaWeb cooperative module makes easy not only the publishing of the homework wording by the tutors but also the delivery of the corresponding report by the groups of students. In this case the students are distributed into *heterogeneous* groups of three students by the teacher. Each team must include students of different characteristics (age, gender and/or specialty) and at least one Erasmus foreign student. Each group has to design and implement an original software development project. Firstly they have to get in touch with the rest the students in the group and then they have to agree on the scope of the software. Students creativity is specifically appreciated in this activity phase. Each group should self-manage to plan the design and implementation of the software project. Before starting the development, all teams must check with the teacher the scope of the program. When

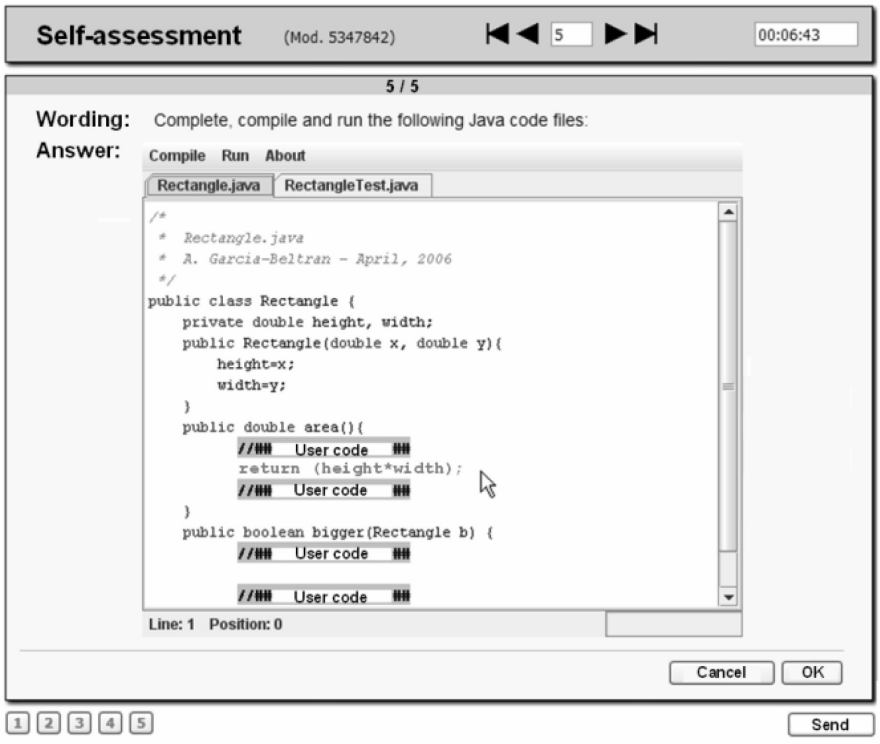

Fig. 2. A Java code question in the student interface in AulaWeb self-assessment system.

finished, the whole application code must be sent to the cooperative module of AulaWeb for evaluation. Group problem solving both in and out of class is a good teaching method since the interactions help many students. Groups are effective since one member of the group often already possesses the programming and/or the *psychomotor* skills. Since psychomotor problems, particularly in communication or speech, can cause both students and practicing engineers' major difficulties, engineering professors should know what resources are available for help. The work will be mark according to the following items: aim and scope of the program, creativity, fellowship and ethical behavior, software design and implementation (to produce clear, reliable, understandable programs), Java programming elements used in the code, no errors execution code readability (comments, indentation...) and oral presentation. Furthermore students are also asked for marking other groups work, primarily creativity and utility. Evaluation is not something that only the professor should do. Students need to practice this skill since they will be expected to be able to evaluate as practicing engineers. This complete activity mark provides the 50% of the student grading.

3. Main results

Since 2005–06 the course is taught using two different methodologies: online (code 9122) and face-to-face format (code 9013). Face-to-face course has allocated two hours a week of class in a computer room and each student has a personal computer with a Java programming environment (JDK + NetBeans IDE) [27–28]. The methodology is eminently practical because every brief explanation of the theoretical contents is followed by the corresponding programming example and its computer validation. The student grade depends on the attendance rate (80% is enough to pass the course). In addition students can voluntarily develop an individual programming work at the end of the term to increase the barely pass grade.

Since 2010–11 both courses (face-to-face and online) were placed in the second semester (February to June) and a proficiency test (not for student grading) is performed at the end of the term in both courses (online and face-to-face) in order to carry out a comparative analysis. Furthermore, the questions in the test are the same every year to ensure consistency of results.

Students are aware of this (non-grading) test from the beginning of the course. The test lasts 30 minutes and consists of 20 short questions developed by the courses teachers. The students of the face-to-face take the exam during the last class (all students can make it together). The online course students take the test after the final teamwork presentation.

Table 1 summarizes the registration data, grades and efficiency rates obtained during the last three courses in the face-to-face course. Efficiency rate is defined as the ratio between the number of passed students and the total number of students enrolled in the course. Therefore this ratio takes in account the drop-out rate.

Table 2 summarizes the registration data, grades and efficiency rates obtained during the last three courses in the online course.

4. A performance comparison

Student learning performance in this online Java course is compared with the traditional face-to-face

Table 1. Students results in the java programming face-to-face course

Academic year	AFG	APT	PS	Total	Efficiency rate
2010–11	7.15	4.40	28	32	87.5%
2011–12	8.32	3.86	34	35	97.1%
2012–13	6.63	3.73	13	16	81.2%
Total			*75*	*83*	*90.36%*

AFG: average final course grade (out of 10). APT: average proficiency test mark (out of 10). PS: number of passed students. Total: number of enrolled students.

Table 2. Students results in the java programming online course

Academic year	AFG	APT	PS	Total	Efficiency rate
2010–11	8.30	5.34	28	28	100%
2011–12	7.48	4.96	28	31	90.3%
2012–13	8.20	5.02	23	24	95.8%
Total			*79*	*83*	*91.8%*

AFG: average final course grade (out of 10), APT: average proficiency test mark (out of 10). PS: number of passed students. Total: number of enrolled students.

Table 3. Summary Statistics for PT (Student Proficiency test mark)

Academic year	PvsD = 1	PvsD = 2
Count	59	81
Average	4.07627	5.01852
Variance	3.89494	3.02778
Standard deviation	1.97356	1.74005
Minimum	1.0	0.0
Maximum	8.5	8.5
Range	7.5	8.5
Stnd. skewness	1.35164	–1.45766
Stnd. kurtosis	–0.895535	0.768502

Java Programming course. As stated before the measure tool for the performance is an *ad-hoc* proficiency test made by both, face-to-face and online, lecturers and taken by the students at the end of the course.

Three variables are used. The mark in the proficiency test, PT (not used for student grading), is the response variable and subject matter. The second one, PVSD, is a qualitative variable that represents the course methodology: 1 means face-to-face and 2 corresponds to online. The third variable represents the academic year the test was performed in. As test questions are the same for all the students and courses and all the academic years the temporal analysis of the last three years is fully coherent. Table 3 shows the PT descriptive statistics for each course methodology:

We can see that the average PT in the online courses is higher (almost one point) than the face-to-face courses one. PT variability is smaller in the online courses. The same results are observed in Figs. 3 (PT histogram) and 4 (PT boxplot):

Experimental Designs technique is used to resolve if PvsD and year are statistically significant on the PT variable. Experimental Designs decompose the variability of response variable (PT) into contributions due to various factors (PvsD and year). The equation for the linear model is given by (1):

$$y_{ijk} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + u_{ijk} \tag{1}$$

where:

$$u_{ijk} \sim NID(0,\sigma) i = 1,2, j = 1,2,3 \text{ and } k = \text{number of replicas} \tag{2}$$

where y_{ijk} is the response (which is PT), μ is a global effect, i.e., the average level of the response, α_i is the main effect of the PvsD. It measures the increase/decrease of the average response for model i with respect to the average level, thus

$$\sum_{i=1}^{I} \alpha_i = 0. \tag{3}$$

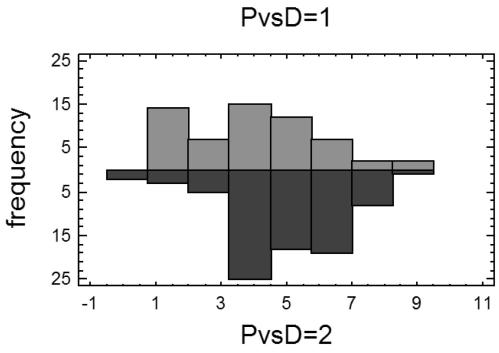

Fig. 3. Histogram of the student proficiency test mark.

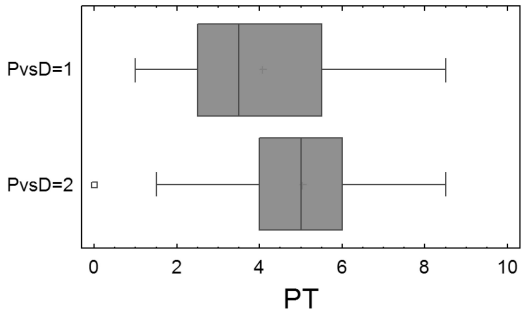

Fig. 4. Boxplot of the student proficiency test mark.

β_j is the main effect of the year. It measures the increase/decrease of average response for length j with respect to the average level, so

$$\sum_{j=1}^{J} \beta_j = 0. \tag{4}$$

$(\alpha\beta)_{ij}$ measures the difference between the expected value of the response and the one computed using a model that does not include the interactions, with

$$\sum_{i=1}^{I} \sum_{j=1}^{J} (\alpha\beta)_{ij} = 0. \tag{5}$$

The inclusion of this term in the model, allows the possibility of the effect of factor "PvsD" on the response variable depending on the level of the other factor, in this case the "year". The random effect, u_{ijk}, includes the effect of all other causes.

The main objective is to test the significance of main effects and second order interactions, so the following tests are performed:

$H_0: \alpha_1 = \alpha_2 = 0$ vs H_1 : any of the α_i different to zero.

$H_0: \beta_1 = \beta_2 = \beta_3 = 0$ vs H_1 : any of the β_j different to zero.

$H_0: \alpha\beta_{11} = \alpha\beta_{12} = \cdots = \alpha\beta_{23} = 0$ vs H_1 : any of the $(\alpha\beta)_{ij}$ different to zero.

Table 4. Multifactor ANOVA—PT. Analysis of Variance for PT—Type III Sums of Squares

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
MAIN EFFECTS					
A:PvsD	30.3407	1	30.3407	8.84	0.0035
B:year	8.04729	2	4.02365	1.17	0.3128
INTERACTIONS					
AB	0.216164	2	0.108082	0.03	0.9690
RESIDUAL	459.868	134	3.43185		
Total (Corrected)	*498.436*	*139*			

All F-ratios are based on the residual mean square error.

Table 5. PT. Analysis of Variance for PT - Type III Sums of Squares

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
MAIN EFFECTS					
A:PvsD	30.5437	1	30.5437	9.03	0.0032
B:year	8.04452	2	4.02226	1.19	0.3077
RESIDUAL	460.084	136	3.38297		
Total (Corrected)	*498.436*	*139*			

All F-ratios are based on the residual mean square error.

The structure of the Analysis of Variance table is shown in Table 4.

Data in Table 4 show that main effect (year) and second order interaction are not are statistically significant since their P-values are bigger than 0.05.

As the F-value in the second-order interaction is less than 1, we can change to a block model. This model studies the effects of the factors on the response variable regardless interaction. The model is similar to the previous one but this model excludes the second order interaction between factors. The analysis of variance for this model is summarized in Table 5.

These data indicate that course methodology is statistically significant but not the academic year. The model diagnosis is suitable and meets the assumptions made in the model.

Figure 5 shows the intervals for the Average Proficiency Test Mark (APT) for each course. There is no overlap between the intervals and therefore there are significant differences between the APT of both courses methodologies. APT in online course is the bigger than in face-to-face one. These results agree with the ANOVA table.

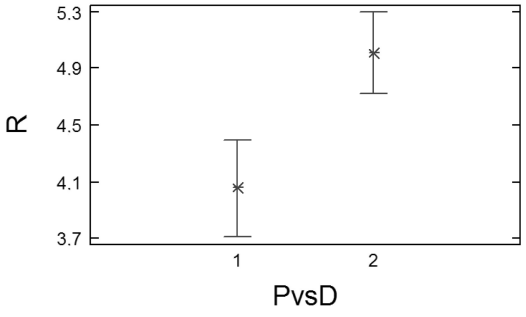

Fig. 5. Means and 95% LSD Intervals for PT.

The comparative analysis results indicate that outcomes are better in the online course.

Three possible reasons may tip the scale against the face-to-face course:

(a) The positive influence of the online course self-assessment system and its immediate feedback on the student performance.
(b) The positive influence of the teamwork activity as a mandatory task to pass the online course.
(c) The negative influence of the student class attendance as a sufficient activity to pass the face-to-face course.
(d) The negative influence of the NetBeans Java IDE (a very powerful tool but difficult to use for non-expert programmers) in the face-to-face classes.

5. Students and teachers opinion

At the end of the term students completed anonymous questionnaires, providing very interesting information and feedback about the courses methodologies. All students generally appreciated the experience of the Java programming courses as part of an engineer's training curriculum and would recommend other students to enroll in both, face-to-face and online, courses.

Flexibility, instant feedback and e-platform ease of use, were seen as one of the major benefits for the online course students. Although primary interest of many students was not computing programming, according to the questionnaires results, 92% of them

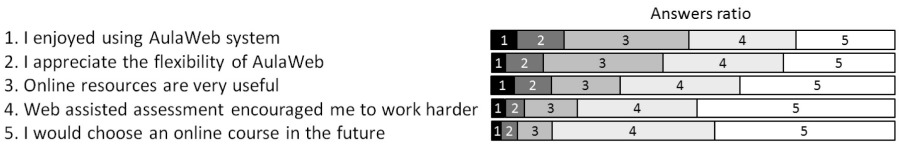

Fig. 6. Summary of the students answers (1, strongly disagree; 2, disagree; 3, neutral; 4, agree and 5, strongly agree) to the online survey at the end of the term for the Java Programming courses (2005–2011).

enjoyed the course and 89% of them thought that the new methodology is very useful for their future life-long learning. On the other hand, some students had problems with their internet connection, the English language or with the working groups. Overall comments were positive, so much so that the majority would be pleased if a similar methodology were used in other future courses.

Figure 6 shows a summary of the students' answers to the most relevant questions of the online questionnaire fulfilled at the end of the term.

Academic staff acceptance is also overwhelmingly positive (100%), 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.

6. Conclusions and future issues

In this paper we have analyzed and compared two methodologies with different activities and teaching-learning tools. These activities and tools make easy that students understand and employ the basic concepts exposed in a programming course and to achieve valuable soft skills (online learning, self-assessment, teamwork. . .) for, especially, life-long learning. On the other hand these activities establish a valuable pool of tools for an easy monitoring and assessment of the student learning process by the course academic staff.

The use of online methodologies is viewed positively by students and tutors. Online students have a great spatial and temporal flexibility to take courses. Furthermore, in this case study, they do not have to install a programming environment locally on their home computers for training and practice purposes. Online students achieve better results than students with the face-to-face methodology. We consider that many typical online activities can be transferred to traditional courses. Depending on the course characteristics, the academic staff can adopt a different methodology/pool of activities approach.

The first future development task is to try to incorporate some online activities to other courses taught in the Department of Control, Electronics and Computer Science. Although our focus was a computer programming course, much of this work should be useful to teachers in other engineering and technical disciplines.

Acknowledgements—This work is partially funded by the División de Informática Industrial of the Universidad Politécnica de Madrid (Spain). The authors would like to acknowledge the implementation support of A. Alonso, J. M. Arranz, P. Avendaño, M. Aza, L. Blanco, S. Campos, D. Cortés, J. A. Criado, F. de Ory, C. Engels, M. Fernández, V. Gámiz, P. García, M. González, J. Granado, T. Hernández, I. Iglesias, J. A. Jaén, A. R. López, D. López, J. A. Martín, M. Martín, R. Martínez, F. J. Mascato, D. Molina, C. Moreno, D. Muñoz, L. M. Pabón, S. Pastor, J. C. Pérez, A. Rodelgo, A. Valero, E. Villalar, H. Waits and C. Zoido.

References

1. M. C. van der Wende, The Bologna Declaration: Enhancing the Transparency and Competitiveness of European Higher Education, *Higher Education in Europe*, **25**(3), 2000, pp. 305–310.
2. T. Hedberg, The impact of the Bologna Declaration on European engineering education, *European Journal of Engineering Education*, **28**(1), 2003, pp. 1–5.
3. T. Saarinen, Quality in the Bologna Process: from competitive edge to quality assurance techniques, *European Journal of Education*, **40**(2), 2005, pp. 189–204.
4. L. Harasim, *Online Education. Perspectives on a New Environment*. Praeger, New York, 1990.
5. R. Mason, Models of Online Courses, *Asynchronous Learning Networks Magazine*, **2**(2), 1998
6. G. G. Smith, D. Ferguson and M. Mieke, Teaching College Courses Online versus Face-to-Face, *Technological Horizons in Education Journal*, **28**(9), 2001, pp. 1–5.
7. K. Benda, A. Bruckman and M. Guzdial, When Life and Learning Do Not Fit: Challenges of Workload and Communication in Introductory Computer Science, *ACM Transactions on Computing Education*, **12**(4), Article 15, November 2012, 38 pages
8. D. G. Langenhorst, *Effectiveness of Online Instruction: Differences in Measured Student Outcomes Online versus Face-to-Face Instruction at the High School Level*, PhD Thesis, Northeastern University, USA, 2012
9. Y. Yang and E. Park, Applying Strategies of Self-Regulation and Self-Efficacy to the Design and Evaluation of Online Learning Programs, *Journal of Educational Technology Systems*, **40** (3), 2012, pp. 323–335.
10. J. Baxter, Who Am I and What Keeps Me Going? Profiling the Distance Learning Student in Higher Education, *International Review of Research in Open and Distance Learning*, **13**(4), 2012, pp. 107–129.
11. A. Driscoll, K. Jicha, A. N. Karl, L. Tichavsky and G. Thompson, Can Online Courses Deliver In-Class Results?: A Comparison of Student Performance and Satisfaction in an Online versus a Face-to-Face Introductory Sociology Course, *Teaching Sociology*, **40**(4), 2012, pp. 312–331.
12. S. Carson, S. Kanchanaraksa, I. Gooding, F. Mulder and R.

Schuwer, Impact of OpenCourseWare Publication on Higher Education Participation and Student Recruitment, *International Review of Research in Open and Distance Learning*, **13**(4), 2012, pp. 19–32.
13. D. Mathew, From Fatigue to Anxiety? Implications for Educational Design in a Web 2.0 World, *Interactive Technology and Smart Education*, **9**(2), 2012, pp. 112–120.
14. H. Abelson, The Creation of OpenCourseWare at MIT, *Journal of Science Education and Technology*, **17**(2), 2008, pp. 164–174.
15. J. Mackness, S. Mak and R. Williams, *The ideals and reality of participating in a MOOC*. Proceedings of the 7th International Conference on Networked Learning 2010. University of Lancaster, Lancaster, (2010) pp. 266–275
16. A. McAuley, B. Stewart, G. Siemens and D. Cormier, *The MOOC model for digital practice*, 2010
17. P. C. Wankat and F. S. Oreovicz, (1993). *Teaching Engineering*, McGraw-Hill, NY, 1993
18. A. García-Beltrán and R. Martínez, AulaWeb: un sistema para la gestión, evaluación y seguimiento de asignaturas, *Industria XXI*, **2**, 2001, pp. 11–16.
19. R. Martínez and A. García-Beltrán, *Manual de AulaWeb*, Sec. Pub. ETSII-UPM, Spain, 2003
20. B. Reinalda and E. Kulesza, *The Bologna Process. Harmonizing Europe's Higher Education*, Barbara Budrich Pub. Opladen & Bloomfiled Hills, 2005
21. C. Fernández, D. Díez, J. Torres and T. Zarraonandía, The cost of learning and teaching Java in the Bologna process, *Proc. of the 2nd Workshop on Methods and Cases in Computing Education*, Barcelona, Spain, 2009, pp. 41–45.
22. ADL, Advanced Distributed Learning, About Sharable Content Object Reference Model (SCORM), http://www.adlnet.gov/Technologies/scorm/default.aspx. Accessed: 30 June 2013
23. A. García-Beltrán, R. Martínez, D. J. Muñoz and J. A. Muñoz-Guijosa, (2007), Implementación de un Módulo de Gestión de Contenidos SCORM en la Plataforma AulaWeb, in *Proc. of SPDECE 2007, Diseño, Evaluación y Desarrollo de Contenidos Educativos Reutilizables*, Bilbao, Spain, 2007, pp. 10-21.
24. A. García-Beltrán and J. M. Arranz, *Programación en Java I*, OpenCourseWare UPM (2009). http://ocw.upm.es/lenguajes-y-sistemas-informaticos/programacion-en-java-i. Accessed: 30 June 2013
25. A. García-Beltrán and J. M. Arranz, *Introducción a la Programación con Java*, Sección de Publicaciones de la ETSII-UPM, Madrid, Spain (2007).
26. A. García-Beltrán, S. Tapia, R. Martínez and J. A. Jaén, Simulator for a Multi Programming Environment for Computer Science Learning and Teaching, *International Journal of Engineering Education*, **25**(2), 2009, pp. 221–227.
27. Java SDK. http://www.oracle.com/technetwork/java/javase/downloads/index.html. Accessed: 30 June 2013
28. NetBeans IDE, https://netbeans.org/. Accessed: 30 June 2013

Ángel García-Beltrán, PhD is an Associate Professor in the Dpt. of Control, Electronics and Computer Science of the Universidad Politécnica de Madrid. He teaches several C, Java Programming and Object Oriented Programming courses and is also active in innovative engineering education methods using the web.

Santiago Tapia, PhD is an Associate Professor in the Dpt. of Control, Electronics and Computer Science of the Universidad Politécnica de Madrid. He teaches C/C++ and Java Programming courses and is active in engineering education using the Internet.

María-Jesús Sánchez-Naranjo, PhD is an Associate Professor in the Dpt. of Organization Engineering, Business Administration and Statistics of the Universidad Politécnica de Madrid. She teaches Statistics, Experimental Design & Regression Models and Time Series Analysis.