Computers & Education 58 (2012) 1–10

Contents lists available at ScienceDirect

Computers & Education

journal homepage: www.elsevier.com/locate/compedu

A distributed system for learning programming on-line

Elena Verdú [a], Luisa M. Regueras [a], María J. Verdú [a,*], José P. Leal [b], Juan P. de Castro [a], Ricardo Queirós [c]

[a] School of Telecommunications Engineering, University of Valladolid, ETSI Telecomunicacion, Paseo Belén 15, 47011 Valladolid, Spain
[b] Department of Computer Science, University of Porto, DCC - R. do Campo Alegre 1021/1055, 4169-007 Porto, Portugal
[c] School of Management and Industrial Studies, Polytechnic Institute of Porto, Rua D. Sancho I, 981, 4480-876 Vila do Conde, Portugal

ARTICLE INFO

Article history:
Received 20 May 2011
Received in revised form
13 July 2011
Accepted 10 August 2011

Keywords:
Distributed learning environments
Interactive learning environments
Programming and programming languages
Teaching/learning strategies

ABSTRACT

Several Web-based on-line judges or on-line programming trainers have been developed in order to allow students to train their programming skills. However, their pedagogical functionalities in the learning of programming have not been clearly defined. EduJudge is a project which aims to integrate the "UVA On-line Judge", an existing on-line programming trainer with an important number of problems and users, into an effective educational environment consisting of the e-learning platform Moodle and the competitive learning tool QUESTOURnament. The result is the EduJudge system which allows teachers to apply different pedagogical approaches using a proven e-learning platform, makes problems easy to search through an effective search engine, and provides an automated evaluation of the solutions submitted to these problems. The final objective is to provide new learning strategies to motivate students and present programming as an easy and attractive challenge. EduJudge has been tried and tested in three algorithms and programming courses in three different Engineering degrees. The students' motivation and satisfaction levels were analysed alongside the effects of the EduJudge system on students' academic outcomes. Results indicate that both students and teachers found that among other multiple benefits the EduJudge system facilitates the learning process. Furthermore, the experiment also showed an improvement in students' academic outcomes. It must be noted that the students' level of satisfaction did not depend on their computer skills or their gender.

1. Introduction

Good programming skills are one of the core competencies that engineering and computer science students are expected to develop. However, students and teachers agree that learning programming is a hard task, especially for first year students (Jenkins, 2002, p. 53). Novice students need to be adequately motivated in order to learn programming in a successful and effective manner.

Computer programming has traditionally been taught and practised as a fundamentally individual activity; however, over the last few years, computer science educators have adopted different collaborative learning practices such as programming in pairs and team projects. Through these collaborative activities, students increase their self-confidence, produce better programs and improve their performance and programming skills (Braught, Eby, & Wahls, 2008; McDowell, Werner, Bullock, & Fernald, 2006; Mckinney & Denton, 2006, p. 138).

The use of competitive learning to motivate students and improve their learning process is well documented too (Burguillo, 2010; Chang, Yang, Yu, & Chan, 2003; Siddiqui, Khan, & Katar, 2008). Competitive learning can lead to a greater engagement of students by arousing their competitive instincts (Anderson, 2006). However, differences in motivation and feelings between winners and losers can exist (Vansteenkiste & Deci, 2003) yet these can be mitigated through different practices: focussing the activity more on learning and enjoyment rather than on winning, clearly defining the evaluation criteria, etc. Verdú and Lorenzo (2010) review the existing literature to report on the advantages and disadvantages of both aforementioned learning strategies, collaborative and competitive, arguing that both learning strategies can be adequate for certain learning contexts. In fact, since not all students are motivated by the same things (Jenkins, 2001, p. 53), it is interesting to be able to define different learning environments: collaborative, competitive or even mixed; since several studies suggest

* Corresponding author. Tel.: +34 983423707; fax: +34 983423667.
E-mail addresses: elever@tel.uva.es (E. Verdú), luireg@tel.uva.es (L.M. Regueras), marver@tel.uva.es (M.J. Verdú), zp@dcc.fc.up.pt (J.P. Leal), jpdecastro@tel.uva.es (J.P. de Castro), ricardo.queiros@eu.ipp.pt (R. Queirós).

0360-1315/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compedu.2011.08.015

that the combination of competitive and collaborative learning also provides high levels of motivation and improves the students' performance (Tauer & Harackiewicz, 2004; Regueras, Verdú, Verdú, & de Castro, 2011).

Besides individual preferences, gender differences have been shown to exist within collaborative and competitive contexts. According to (McDowell et al., 2006), collaborative activities may help to increase women's representation in the computer science area, a field where women have a low participation. On the other hand, different studies suggest that men are more motivated in competitive environments while female motivation is unaffected by competition (Manne, 2000, p. 129; Van Vugt, De Cremer, & Janssen, 2007). In the context of teaching programming, competition is a very important element since the combination of contests with automated assessment provides the educational community with an effective and efficient learning tool. Moreover, for programming students, the competitions are fun (Manne, 2000, p. 129).

Programming contests with automated assessment have become popular activities for training programming skills (Trotman & Handley, 2008). There are several web-based on-line judges where students can download programming problems and submit their solutions for automatic judging; one of the most popular is the "UVA On-line Judge" (http://uva.onlinejudge.org/). The UVA On-line Judge is an on-line programming trainer created in 1995 with the aim of training users as they participate in world-wide programming competitions. Currently, this judge hosts several hundred problems from regional and world competition finals dating back many years and has an important number of users, as approximately 80% of the world finalists have been using this on-line judge since its creation in 1995 (Revilla, Manzoor, & Liu, 2008; Skiena & Revilla, 2003). This system is very interesting not just for students, but also for teachers; however, it has several limitations in order to be incorporated into official programming courses. In this context, the EduJudge project arises to satisfy the users' demand of a greater pedagogic character for the judge and, in this way, to facilitate its use as a regular activity in official courses. This is a project funded with the support from the European Commission whose aim is to integrate the "UVA On-line Judge" into an effective educational environment: the Moodle Learning Management System and the QUESTOURnament competitive learning tool. The main result of the project is the EduJudge system, which is a distributed system that is composed of three subsystems: an evaluation server, a learning objects repository and a user interface.

This paper discusses the students' satisfaction and their academic performance after using this competitive educational on-line tool as part of their programming courses. Section 2 introduces the major issues related to learning programming on-line and advances this study's research problems and formulates its hypotheses. An overview of the EduJudge system is also provided in Section 2. The subsequent section presents a case study on the use of EduJudge, the results of which are analysed in Section 4. Finally, the conclusions are presented.

2. Learning programming on-line

Rongas, Kaarna, and Kalviainen (2004) classify introductory programming learning tools into four types: integrated development interfaces, visualisation tools, virtual learning environments, and submission systems. It is difficult to find a tool that successfully incorporates the advantages from each type. For programming learning it is important to provide, at least, a combination of a virtual learning environment and a submission system. The former is the only type that provides support for problem solving and high support for individual study and self-teaching. With the latter, most submission systems support a variety of assignment and evaluation formats. In conjunction they provide a system for automatic assessment and feedback. The goal of EduJudge was the integration of a submission system with a virtual learning environment of reference. The resulting system allows for the submission, management and automatic evaluation of programming exercises and the development of competitions as part of a Moodle course.

2.1. Related work

Providing instant support to students is critical for the teaching computer programming successfully (Wang & Wong, 2008). Receiving feedback is very important for the acquisition of knowledge regardless of the particular learning strategy (Mory, 2004). Furthermore, immediate feedback motivates students (Daly, 1999, p. 155; Truong, 2007). However, providing individualised feedback to all students is time-consuming for teachers and frequently involves a time delay. Consequently, an important number of on-line submission systems that support the automated evaluation of programming problems have been proposed and successfully used in programming courses: WebTasks (Roessling & Hartte, 2008, p. 363), Oto (Tremblay, Guerin, Pons, & Salah, 2008), CourseMarker (Higgins, Gray, Symeonidis, & Tsintsifas, 2005), RoboProf (Daly, 1999, p. 155), Marmoset (Spacco, Hovemeyer, Pugh, Emad Hollingsworth & Padua-Perez, 2006), Web-CAT (Edwards & Perez-Quinones, 2007), Viope (Vihtonen & Ageenko, 2002, p. 371), PASS (Wang & Wong, 2008), BOSS (Heng, Joy, Boyatt, & Griffiths, 2005), Mooshak, etc. All these tools allow students to submit their solutions to programming problems and to receive instant feedback. As a result students can practise and improve their programming skills independently.

Although grading and providing meaningful feedback is a popular method to engage programming students, some researchers offer other features to improve the learning process. WebTasks (Roessling & Hartte, 2008, p. 363), a web-based training platform where novice students learn how to program Java, provides a form of peer-review as well as feedback and resubmission. If a student submits an incorrect solution, he/she can resubmit it; whereas if the solution is accepted, the student can see all the other accepted solutions and comment on them. A similar feature is also found in QUESTOURnament (and therefore, in EduJudge) where students can read all submissions and their corresponding evaluations. Moreover, students may propose problems or challenges to their class-mates during the competition and assess the corresponding answers. With this feature students learn from their colleagues' answers, as well as from the assessment process itself whilst preserving their anonymity. These peer-to-peer interactions help foster a feeling of connection among the members of the course (Mory, 2004), which contributes to mitigate the isolation that may be felt in on-line learning environments. Besides, peer-reviews allow students to deepen and broaden their knowledge, to enhance problem solving skills and to develop transversal competencies that are commonly required in professional situations (Rice, Beg, Waters, Bokreta, & Santiago-Aviles, 1999; Van der Pol, Van den Berg, Admiraal, & Simons, 2008; Yang, 2010). Moreover, students tend to perform better when they know that their work will be viewed by their class-mates (Hislop, 1999; Regueras, Verdú, Pérez, De Castro, & Verdú, 2007).

While some researchers propose tools for assisting teachers in grading such as BOSS/BOSS2 (Heng et al., 2005), other authors focus on the automatic feedback and grading. This is the case of: RoboProf (Daly, 1999, p. 155), a system which provides simple feedback but allows teachers to monitor students' progress in order to provide more detailed feedback on concepts that prove problematic; CourseMarker

(Higgins et al., 2005), a customisable and extensible evaluation system based on Ceilidh (Higgins, Hegazy, Symeonidis, & Tsintsifas, 2003); Oto, which provides instant feedback to students but also offers feedback prior to the final submission (Tremblay et al., 2008); and Web-CAT, a system which assesses the validity and the completeness of the student's test cases and generates concrete and tailored feedback (Edwards & Perez-Quinones, 2007).

The combination of teacher support and meaningful automated feedback can be also found in the literature. For example, Marmoset (Spacco et al., 2006) provides students with automated feedback in the form of incentives prior to the final submission in order to instill good programming habits (starting early and thinking critically about complex input cases) and helps teachers understand students' work habits.

Besides grading and meaningful feedback, other characteristics can be included to improve the learning process, such as providing a problem database with different levels of difficulty. It has been shown that this can improve students' performance and satisfaction levels (Wang & Wong, 2008). Students with lower computer skills can begin by solving easier problems in order to learn progressively and to stay motivated to solve the harder problems later (Lee & Heyworth, 2000). This is very important since, according to Wiedenbeck, LaBelle, and Kain (2004, p. 97), one of the factors that has a greater influence on the students' success in learning programming is previous computer experience.

Another interesting feature in programming learning tools is the possibility to organise competitions, usually combined with an automated evaluation system. This is the case of Mooshak, a Web-based programming contest management system. To increase competitiveness the system shows a ranking of students according to the number of problems solved. The Edujudge system improves this type of competitive system by allowing not only individual but also team competitions. This mixed environment is more useful for learning programming and for acquiring teamwork skills at the same time (Manne, 2000, p. 129).

Moreover, some of the aforementioned tools provide extra features to teachers, such as messaging systems, forums and even access to course materials. However, one of the main drawbacks of these systems is that they are not integrated into common e-learning platforms. This is the case of Viope (Carver & Henderson, 2006, p. 9; Vihtonen & Ageenko, 2002, p. 371), a standalone solution for programming learning and automated evaluation by means of a complete interactive learning environment. It is very important that the learning environment includes as many components of the learning process as possible besides the evaluation of programming exercises. Other functionalities include course documentation, tutorship, management of users and grades. For example, Guerreiro and Georgouli (2008) combine the use of Mooshak and the Moodle platform in order to overcome this limitation, but without a true integration of both tools.

Unlike all the above mentioned tools, EduJudge integrates a submission system with automated evaluation into a widely-used and well-known virtual learning environment: Moodle. The Moodle platform incorporates an innovative module for organising competitions, the QUESTOURnament tool, and many other learning tools for collaborative and individual learning, making it possible to apply different learning strategies. The EduJudge system combines automated feedback with teacher's feedback, which is provided through the Moodle platform. EduJudge also provides a problem database with different levels of difficulty. Currently, this repository contains a large variety of programming problems that can be searched by keyword, author, type, instruction level, difficulty level and language. Unlike other learning tools such as Viope, which works only with very short programming exercises and it is aimed only for introductory programming courses, EduJudge can work with very complex problems and, therefore, can also be used for higher levels of programming competencies whose development requires more difficult programming assignments.

2.2. An overview of the EduJudge system

EduJudge is a distributed system for learning programming; it is composed of three main subsystems: an evaluation server, a learning objects repository and a user interface. Fig. 1 shows the interactions among the different components and users of the EduJudge system.

The evaluation server is an evolution of the programming trainer "UVA On-line Judge". The new evaluation engine is able to provide a nuanced evaluation (instead of a binary correct/incorrect evaluation) and to evaluate different types of programming problems: problems having a single test case, problems having several test cases, and interactive problems in which the solution must interact with another program. It also supports a wide variety of programming languages such as Pascal, Java, C and C++.

The repository of problems, called crimsonHex, improves the accessibility to and usability of algorithm problems. The purpose of this interoperable Learning Object Repository (LOR) is to store the programming problems as Learning Objects (LOs) and supply them to the Learning Management System and to the evaluation server in a standards compliant way. Currently, this repository contains a large variety of programming problems that can be searched by keyword, author, type, instruction level, difficulty level and language.

Finally, the EduJudge User Interface (UI) consists of a set of plugins and modules for Moodle supporting different pedagogical techniques in combination with the automatic evaluation of programming exercises retrieved from the crimsonHex repository. Teachers can use programming exercises either as part of a Moodle quiz or as part of a contest driven by the QUESTOURnament activity. In fact, the EduJudge basic functionality has been implemented as a new question type suitable to be used with any Moodle activity that uses the Question Engine, the core component for the management of questions and answers within Moodle. By using quizzes teachers can evaluate the progress of their students. Quizzes can also be used for assessment in self-managed learning. The QUESTOURnament activity has been designed to encourage students to compete among themselves attempting to solve a set of timed quizzes. It engages students in a more interactive, competitive and motivating environment.

The following are the steps for supporting a programming contest with EduJudge in an educational environment:

1. The teacher creates a QUESTOURnament activity and adds challenges to the contest. Each challenge can be composed of one or several programming problems, which must be retrieved from the LOR the first time that they are used.
2. Students answer the challenges and get automated feedback from the evaluation server (the on-line judge).
3. The competition goes on: the teacher supervises the whole process and students check their ranking and the teacher's feedback.

The above steps are the basics for a typical and simple configuration of QUESTOURnament yet teachers can decide to enrich the process, for example allowing students to propose challenges for the contest and assessing the answers of their classmates.

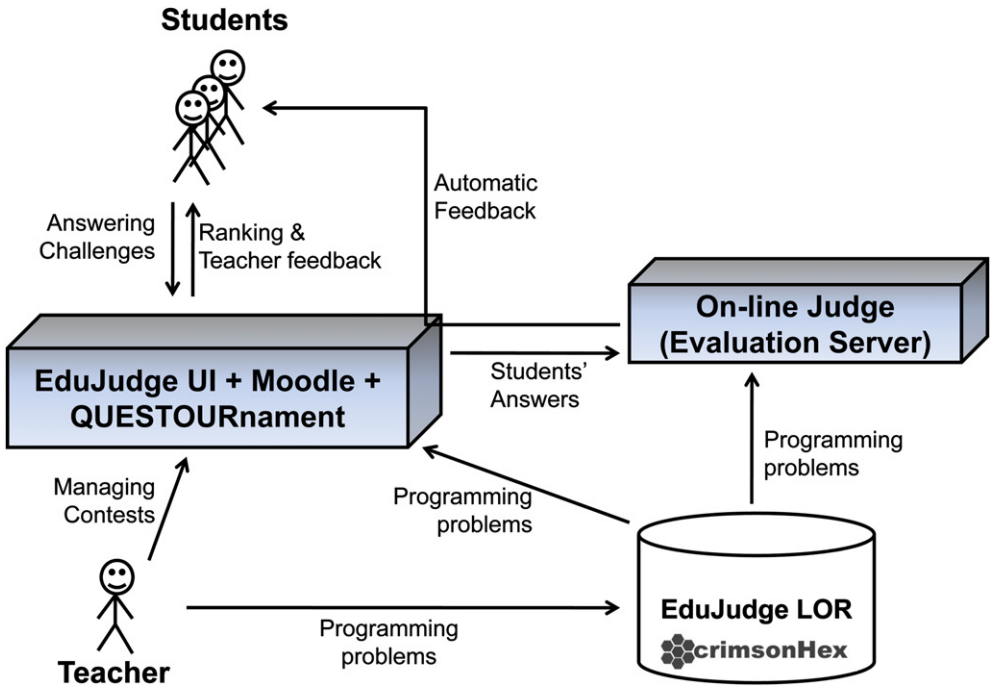

Fig. 1. Structure and interactions of the EduJudge system.

Both teachers and students can use EduJudge in order to increase motivation and performance. Teachers can reuse and share problems, and organise programming contests in their Moodle courses. Students can access an important number of programming problems according to their profile and their abilities, submit their programs as solutions and automatically receive feedback from the system. In addition, once a solution has been evaluated by the automated evaluation server, teachers can add further comments or change the grade if they do not agree with the mark given by the system.

In summary, the complete EduJudge system provides:

- Programming problems with an adequate and standard structure, the solutions to which can be automatically evaluated by an on-line server.
- A search engine so that the problems are more readily accessible for their potential users.
- An opportunity to apply different pedagogical approaches using the proven e-learning platform Moodle.

More details and examples about the operation and the characteristics of the EduJudge system are available in previous publications (De Castro, Pérez, Regueras, & Verdú, 2010; De Castro, Verdú, Regueras, Pérez, & Verdú, 2009).

2.3. Challenges, issues and research hypotheses

Most students think that programming is boring and difficult (Jenkins, 2002, p. 53). Hence, one of the most important problems about teaching programming is how to motivate students. Students who do not have an intrinsic motivation will hardly succeed (Gomes & Mendes, 2007). Therefore, tools and learning environments used in introductory programming courses should appeal to students by including interesting activities. Different studies show that automated assessment promotes the interest of students in learning programming (Fernandez, 2011; Trotman & Handley, 2008). Also, there are pedagogical studies indicating that competitive learning increases the motivation, interest and satisfaction of students (Anderson, 2006; Burguillo, 2010; Chang et al., 2003; Siddiqui et al., 2008). The EduJudge system includes both features: automated assessment and the possibility of organising competitions. Therefore, this study proposes firstly the following research question about the students' motivation and satisfaction through the use of the EduJudge system: what is the level of satisfaction of students who use the Edujudge system?

Tools for learning programming should also support individual study, provide progress reports and allow for different learning styles (Jenkins, 2002, p. 53; Rongas et al., 2004, p. 678). Moreover, programming learning should be practical and concrete (Jenkins, 2001, p. 53; Lahtinen, Ala-Mutka, & Jarvinen, 2005). It is very important to give students the opportunity for ample practice so they may solve programming exercises by themselves. In this context, one of the main advantages of the submission system is that students can directly apply the concepts as they problem-solve. Different studies show that automated evaluation systems significantly improve the performance of students (Fernandez, 2011). Moreover, the development of competitive learning activities together with a component of peer review reduces procrastination and improves the learning process. Competition has been used in several programming teaching contexts with very good results (Fernandez, 2011; Guerreiro & Georgouli, 2008; Lawrence, 2004). Thus, the following hypothesis is formulated about the effectiveness of the EduJudge system in the improvement of academic performance:

H1. The students who use the EduJudge system will obtain higher final exam scores (and thus, improve their academic performance) than those students who do not use it.

Some studies identify several factors (such as previous computer experience and gender) as significant predictors of the performance and satisfaction of novice programming students (Daly & Horgan, 2004; Pillay & Jugoo, 2005; Wiedenbeck et al., 2004, p. 97). Most research provides evidence that previous computer experience has a positive effect on success in the learning of programming (Byrne & Lyons, 2001, p. 49). The influence of gender is not so evident since different studies show that gender differences are not always significant (Lau & Yuen, 2009; Murphy et al., 2006). Moreover, both factors overlap since female students usually have less previous computer experience than their male classmates due to the gender gap existing in computer use (Murphy et al., 2006; Vekiri & Chronaki, 2008). Thus, prior computer skills and gender differences are also considered in the following hypotheses about the differences in programming learning with the EduJudge system:

H2. The level of satisfaction of students with higher computer skills will be higher than that of the students with lower computer skills when using the EduJudge system.

H3. The level of satisfaction of women will be different than that of men when using the EduJudge system.

3. Case study: engineering courses

This section provides a description of the main features of the courses in which the experiment was carried out. Then, the experiment's design is described and the premises of the study are explained. Also, a description of the data collected and the instruments used to evaluate the experience is given.

3.1. The educational context

During the academic year 2009–2010, EduJudge was used and tested in three algorithms and programming courses at the Polytechnic Institute of Porto: 'Information Structuring, Analysis and Processes Systematisation' (degrees in Biomedical Engineering and in Industrial Management Engineering) and 'Algorithmic and Programming' (degree in Mechanical Engineering). These courses are integrated into the curricula of engineering degrees not focused on computer science and aim to introduce students to programming concepts during the first year of these degrees. They have a strong algorithmic component to help students analyse problems and plan the steps for solving them. The learning objectives of the courses are:

- Application of information analysis methods and systematisation of processes.
- Enhancement of the students' problem-solving skills.
- Development of computer applications using high level programming languages, such as Pascal and C#. The EduJudge system has been used to help students to reach these objectives.

The EduJudge system has been used to help students to reach these objectives.

3.2. The experiment

The experiment took place during April and May of 2010. Students' motivation and satisfaction were analysed by means of in situ observation and satisfaction questionnaires. In addition it was necessary to analyse the effects of the EduJudge system on students' academic outcomes. For this, the experimental research method was applied (Oncu & Cakir, 2011) and experimental and control groups were established in order to identify a relationship between variables. However, a completely randomised design was not possible and a static group comparison design (Fraenkel & Wallen, 2000) was used as teachers decided not to divide their students randomly into two groups as they felt it to be unfair for those students falling in the control group. Hence, in order to evaluate the experience all students taking the three engineering courses during the academic year 2009–2010 were considered as belonging to the experimental group. Their academic results would then be compared with those obtained by the students taking the same courses during the subsequent academic year (course 2010–2011) when the system would not be used. The following two premises were established: the only difference between the two academic years would be the use of this system in the first and not the second and there should not be significant differences in the students' demographic make-up from one academic year to another.

A total of 53 students with different gender and previous computer skills enrolled in the three courses participating in the experiment.

Training was provided mainly on-site so that the students were assisted by their teachers while solving the exercises. The following steps were carried out in preparation for the experiment:

1. Introduction of QUESTOURnament to the teachers involved in the experiment.
2. Definition of an experimentation plan (e.g. documentation, schedule…).
3. Creation of the learning objects: four Pascal programming exercises.
4. Creation of a tutorial to give to the students at the beginning of the experiment.

The experiment started with a brief explanation of the system and a tutorial given to the students. It was followed by the students' resolution of the exercises while assisted by their teachers.

The following year, during the academic year 2010–2011, a total of 56 students enrolled in the three courses. There were no demographic differences between these students and the students in the experimental group. Although it could be assumed that the population of the courses was almost randomly formed, strictly the experiment should be called a quasi-experiment.

Table 1
Comparative of academic outcomes.

Course	Academic Year 2009–2010 (EduJudge)			Academic Year 2010–2011 (no-EduJudge)			*T*-Test	
	n	M	SD	n	M	SD	T	*p*
Biomedical Engineering	19	11.15	1.23	16	9.45	3.09	2.017	0.030*
Industrial Management Engineering	9	11.67	0.82	13	8.83	4.42	2.900	0.003**
Mechanical Engineering	25	12.19	2.06	27	9.92	4.07	3.145	0.001**

*Results are significantly different at $p < 0.05$ (*T*-Test).
**Results are significantly different at $p < 0.01$ (*T*-Test).

3.3. Instruments and data collection

Two instruments were used in this study: a) the students' final exam grades in the three courses for both academic years, and b) a survey, which measures students' satisfaction with their learning experience using the EduJudge system. This survey included several items about the EduJudge components and the system as a whole along with general demographic data. Specifically, the survey was composed of three different parts:

- Personal data for statistics: age, gender, computer skills…
- Five-score Likert-type scale items, which ranged from "Strongly Disagree" to "Strongly Agree", for analysing the level of satisfaction with the EduJudge system.
- Yes/no items for assessing both the quality of the problems posed and the functionality of the on-line Judge.

Data from the survey was collected on-line when the courses finished. The survey was completed by all the teachers and by 33 students (about 62%). Although an attrition in the students sample is observed, it has been checked that respondents are equivalent to the whole sample in gender, computer skills and academic performance. Teachers who participated in the study think that the cause of attrition is that the survey was available when the course had already finished and, therefore, some students were inactive or could not be found. Thus, it could be assumed that the attrition has not biased the results.

The data collected on students' outcomes (final exam grades) were analysed for group comparison using the Student *T*-Test. This statistic indicates whether the means of two groups are statistically different from each other in order to be able to compare them. In addition, in order to check whether students' satisfaction with the EduJudge system differed according to gender and computer skills and to investigate whether there was any interaction among these variables, a two-way analysis of variance (ANOVA) was also conducted.

4. Results and discussion

In order to assess the pedagogical performance of the system, a correlation between students' academic outcomes and their use of the system was first analysed. Then this section will discuss the students' degree of satisfaction with the use of the EduJudge system and its different components based on the survey data. The purpose of this analysis is to validate the usefulness of the system, since several studies (Donohue & Wong, 1997; Levy, 2007) suggest that students' satisfaction and motivation are important factors in measuring the success or effectiveness of the e-learning process. The analysis of results is done in general terms and also answering the research question and testing the hypotheses formulated in Section 2.3.

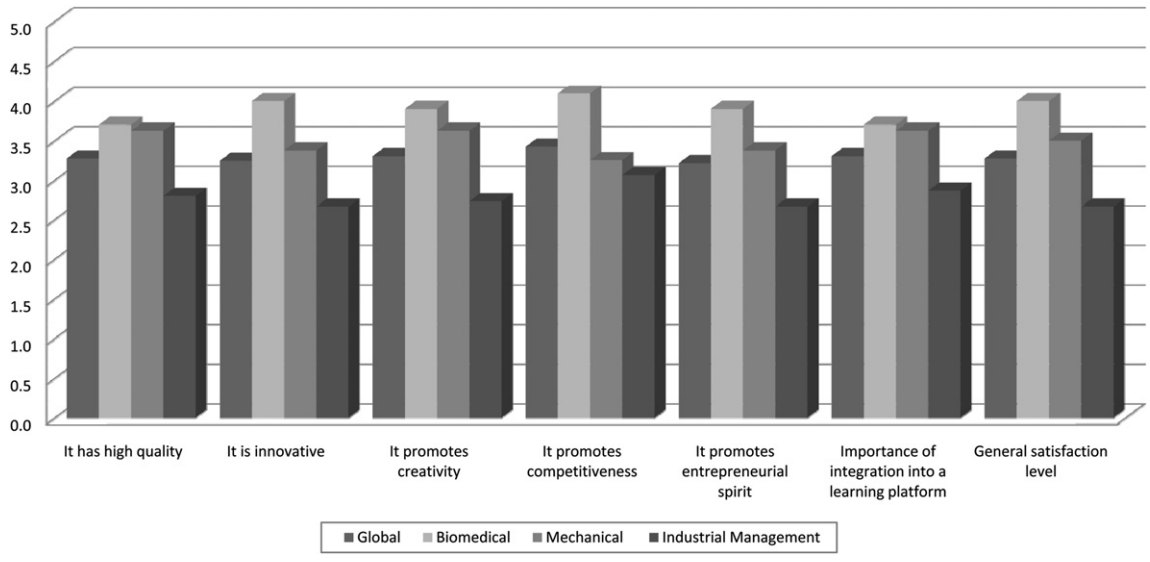

Fig. 2. Results of students' opinion about the EduJudge system.

Table 2
Mean (M) and Standard deviation (SD) for ITEMS ABOUT students' opinion on the EduJudge system.

Items	M	SD
Satisfaction level	3.273	0.761
Importance of integration into a learning platform	3.303	0.770
It has high quality	3.273	0.876
It is innovative	3.242	1.032
It promotes creativity	3.303	1.045
It promotes competitiveness	3.424	1.062
It promotes entrepreneurial spirit	3.212	0.960

4.1. Analysis of the academic outcomes

Table 1 shows the mean (M) and the standard deviation (SD) of the score obtained by students in the three courses for the two different academic years considered. An improvement in the final score can be observed when this system is applied to programming learning. According to the results in Table 1, students who used the EduJudge system achieved significantly better academic outcomes than those who did not use it across all three courses. This result indicates that the hypothesis H1 is supported.

A deeper analysis shows an interesting find. The standard deviation of the scores in the EduJudge group is smaller than the one in the control group. This fact indicates that the score is less dispersed in the EduJudge group. Thus demonstrating a more homogeneous improvement in students' academic performance is obtained when all the pedagogical aspects described in Section 2 are combined in an on-line tool, adapting to different preferences and profiles.

4.2. Analysis of the satisfaction

Once the results of the surveys were available it was important to study their reliability. Cronbach's alpha was tested and the calculated alpha value for the satisfaction with the EduJudge system was 0.95, indicating very high reliability (Straub, 1989).

In general terms, the survey data shows the experience was evaluated positively by students. Fig. 2 summarises the main results for the sample as a whole and individually for each course, where bars represent the average score assigned to each item (5 being the maximum score). They liked learning through this system and positively assessed the integration of programming competitions into a learning platform like Moodle. Besides, students think that the EduJudge system helped to promote creativity, employability and competitiveness. Most students reported a high satisfaction with the EduJudge system. The students from the Industrial Management degree presented less satisfaction than their colleagues; probably because they were affected by a technical failure during the experiment that delayed feedback, according to the teachers' opinion and observation. This shows the importance of appropriate and timely feedback.

Table 2 shows a more detailed statistical study (with the mean and the standard deviation) of the different items in the opinion survey. In summary, and answering the research question "Which is the level of satisfaction of the students who use the EduJudge system?", the results indicate that the students' level of satisfaction with the EduJudge system was 3.273.

The survey also included several questions about the type of participation in the competitions (anonymous, public, individual or in group), but students had no preference. This result differs from other studies, where students preferred to participate publicly (Fernandez, 2011) or anonymously (Regueras et al., 2009). The Judge's behaviour was also analysed. According to the students' opinion, feedback was fast (67%) but lacked detail as less than half of the students (45%) agreed that the system was helpful in searching for errors when the submission was not correct.

Other elements evaluated were the exercises. Most students thought that the proposed exercises were useful (79%), attractive (76%) and provided statements that facilitated their comprehension (70%). Thus, from this data, it can be deduced that the quality of the problems was satisfactory. It is interesting to highlight a student's comment who indicated that the problems could have been more difficult in order to have been made to work harder. This demonstrates the system's effect on students' motivation but also the need to go further. EduJudge users asked for a more flexible system that adapts more to their needs, features and knowledge levels. Offering students programming problems with difficulty levels according to their skills and capabilities would increase the efficiency and motivation of students (Lilley, Barker, & Britton, 2004). Therefore, to turn the EduJudge system into an *adaptive problem sequencing system* is a challenge to explore.

Similarly, the teachers were very satisfied with the EduJudge system. They felt that the system has important advantages since it integrates the programming problems submission and evaluation system into a common framework that supports all the different activities of the learning process. However, just as the students, they also thought that the feedback provided when a solution is incorrect should be more detailed. Teachers had to complete the feedback provided by the system with their own feedback in order to guide students to correct their mistakes, but this feedback usually is not fast enough. Automated feedback could be improved in order to offer instant feedback on the most common mistakes and, thus, allow teachers to devote more time to offer tailored feedback.

Table 3
Two-way anova for the student's satisfaction with the EduJudge system.

	Sum of squares	Df	Mean square	F	*p*
Gender	4.819	1	4.819	4.78	0.037
Computer skills	3.634	2	1.817	1.8	0.184
Gender × computer skills	0.378	2	0.189	0.19	0.828
Error	28.229	28	1.008		
Total	37.059	33			

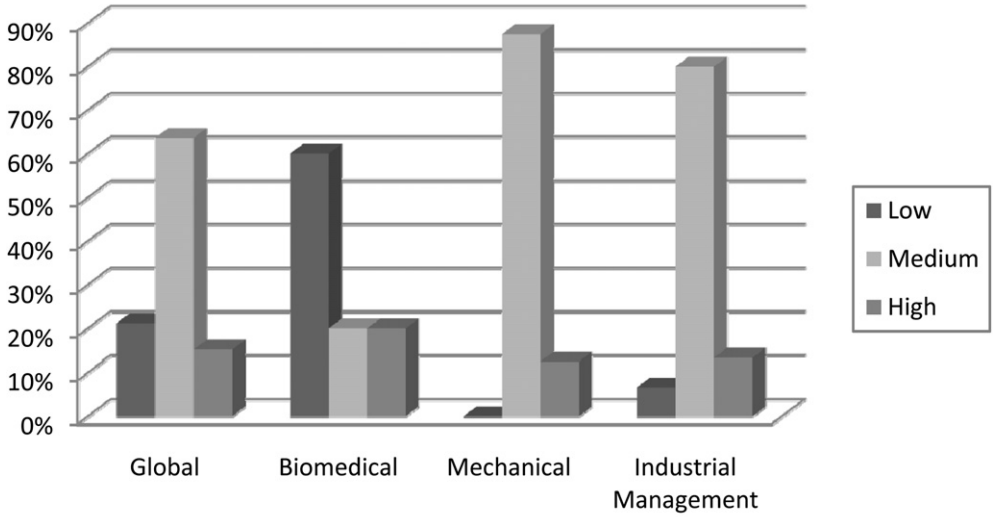

Fig. 3. Students' profile: self-perception of computer skills.

Finally, the different hypotheses proposed about the relationship between the level of satisfaction and gender and computer skills had to be validated. Results of Table 3 indicate that students' satisfaction with the EduJudge system was not different in relation to gender ($F = 4.78$, $p > 0.05$), computer skills ($F = 1.8$, $p > 0.05$) or the interaction of both ($F = 0.19$, $p > 0.05$). Therefore, no differences are found in satisfaction between male and female students and among students with different computer skills. Thus, the hypotheses H2 and H3 are not supported. In fact, the students who indicated most satisfaction were those from Biomedical Engineering degree (see Fig. 2) despite having less computer skills (see Fig. 3). This is a very interesting result, because it could mean that introducing competitive tools and on-line judges into a pedagogical environment makes it possible to engage more students in programming learning than just those more advanced students who typically participate in on-line programming competitions.

5. Conclusions

This paper reports on a study using the EduJudge system for the programming teaching-learning process in three engineering courses. Firstly, students like the EduJudge system since they regard it as useful, facilitating the learning process and promoting creativity and competitiveness. Similarly, teachers think that there are a lot of benefits in using an e-learning competitive tool based on ICTs such as the EduJudge system: the integration of several components in order to achieve a common goal, the possibility of managing the execution of the exercises in several contexts (curricular context: assignments, tests; non-formal learning context: programming competitions), or the availability of the system, among others. One of the most interesting results is that, contrary to what had been proposed as a hypothesis, the level of students' satisfaction does not depend on their computer skills levels. Therefore, the EduJudge system could be suitable for a variety of programming training environments and not just in training for high-level competitions, as typically occurs with isolated on-line judges.

Another find is that according to teachers and students' opinion automatic feedback from the on-line judge should be improved. As well as providing a fast and more detailed feedback when the solution is incorrect, the system could offer more detailed feedback for improving the quality of a program, for example, if it is poorly coded or too complex. It could also offer alternative solutions as suggested by (Truong, 2007). This improvement could complement the current feature of the EduJudge system which allows students to read their classmates' answers once the competition has finished.

EduJudge provides a problem database with different levels of difficulty that has shown to be useful in increasing programming comprehension, as shown in the survey results and academic outcomes. At present, teachers must select the problems they present to students according to their selected difficulty level. An interesting future research line could be to automate this task by turning the Edujudge system into an adaptive learning system. Since problems that prove too difficult, or too easy, could decrease students' motivation, adaptive problem sequencing would provide a more efficient and effective learning (Wauters, Desmet, & Van den Noortgate, 2010).

Regarding gender differences, results show no differences between male and female satisfaction levels. This conclusion is consistent with (Murphy et al., 2006), whose results show that female students, despite having significantly less pre-college programming experience, succeed at comparable rates as their male counterparts.

Moreover, the results of this study indicate that the use of the EduJudge system has important effects on the students' academic outcomes. The students who used the system obtained better final grades. Therefore, the results hereby presented suggest that the EduJudge system can support effective learning strategies for programming learning based on competition and automated assessment in official courses.

Finally it should be indicated that due to the limitations of the groups assigned in the experiment it is not possible to generalise its results. Specifically, both the number of students participating and the number of problems proposed during this experiment were not too big. Furthermore, the division in control and experimental groups was not really random. However, the positive results of this study can be used as a basis for further investigation about the impact of the EduJudge system on programming learning on university students. Finally, a qualitative research that includes observation through time could achieve alternative interpretations of the data and, thus, a richer discussion.

Acknowledgements

The EduJudge project (ref. 135221-LLP-1-2007-1-ES-KA3-KA3MP) has been funded with support from the European Commission. This publication reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

References

Anderson, J. R. (2006). On cooperative and competitive learning in the management classroom. *Mountain Plains Journal of Business and Economics, Pedagogy, 7*, 1–10.
Braught, G., Eby, L. M., & Wahls, T. (2008). The effects of pair-programming on individual programming skill. *ACM SIGSE Bulletin, 40*(1), 200–204.
Burguillo, J. C. (2010). Using game theory and competition-based learning to stimulate student motivation and performance. *Computers & Education, 55*(2), 566–575.
Byrne, P., & Lyons, G. (2001). The effect of student attributes on success in programming. In *Proceedings of the 6th annual conference on innovation and technology in computer science education* (p. 4952).
Carver, J., & Henderson, L. (2006). Viope as a tool for teaching introductory programming: an empirical investigation. In *Proceedings of the 19th conference on software engineering education & training* (pp. 9–16).
Chang, L. J., Yang, J. C., Yu, F. Y., & Chan, T. W. (2003). Development and evaluation of multiple competitive activities in a synchronous quiz game system. *Journal of Innovations in Education and Training International, 40*(1), 16–26.
Daly, C. (1999). RoboProf and an introductory computer programming course. In *Proceedings of the 4th Annual SIGCSE/SIGCUE conference on innovation and technology in computer science Education* (pp. 155–158).
Daly, C., & Horgan, J. M. (2004). An automated learning system for Java programming. *IEEE Transactions on Education, 47*(1), 10–17.
De Castro, J. P., Pérez, M. A., Regueras, L. M., & Verdú,, M. J. (2010). *EduJudge system handbook. How to organize programming competitions in Moodle courses*. Madrid: Sello Editoria.
De Castro, J. P., Verdú, E., Regueras, L. M., Pérez, M. A., & Verdú, M. J. (2009). A proposal of user interface for a distributed asynchronous remote evaluation system: an evolution of the QUESTOURnament tool. In *Proceedings of the 9th IEEE international conference on advanced learning technologies* (pp. 75–77).
Donohue, T. L., & Wong, E. H. (1997). Achievement motivation and college satisfaction in traditional and nontraditional students. *Education, 118*(2), 237–244.
Edwards, S. H., & Perez-Quinones, M. A. (2007). Experiences using test-driven development with an automated grader. *Journal of Computing in Small Colleges, 22*(3), 44–50.
Fernandez, J. L. (2011). Automated assessment in a programming tools course. *IEEE Transactions on Education*, . doi:10.1109/TE.2010.2098442.
Fraenkel, J. R., & Wallen, N. E. (2000). *How to design & evaluate research in education* (4th ed.). Boston: McGraw-Hill.
Gomes, A., & Mendes, A.J. (2007). Learning to program - difficulties and solutions. In *Proceedings of the international conference on engineering education*. <http://icee2007.dei.uc.pt/proceedings/papers/411.pdf> [Last Accessed 11.04.11].
Guerreiro, P., & Georgouli, K. (2008). Enhancing elementary programming courses using E-learning with a competitive attitude. *International Journal of Internet Education (IJIE), 10*, 38.
Heng, P.-S., Joy, M. S., Boyatt, R. C., & Griffiths, N. (2005). *Evaluation of the BOSS online submission and assessment system*. Technical Report. Coventry: University of Warwick.
Higgins, C. A., Gray, G., Symeonidis, P., & Tsintsifas, A. (2005). Automated assessment and experiences of teaching programming. *ACM Journal of Educational Resources in Computing, 5*(3), 1–21.
Higgins, C., Hegazy, T., Symeonidis, P., & Tsintsifas, A. (2003). The CourseMarker CBA system: improvements over Ceilidh. *Education and Information Technologies, 8*(3), 287–304.
Hislop, G. W. (1999). Anytime, anyplace learning in an online graduate professional degree program. *Group Decision and Negotiation, 8*, 385–390.
Jenkins, T. (2001). The motivation of students of programming. In *Proceedings of the 6th conference on innovation and technology in computer science education* (pp. 53–56).
Jenkins, T. (2002). On the difficulty of learning to program. In *Proceedings of the 3rd annual LTSN_ICS conference* (pp. 53–58).
Lahtinen, E., Ala-Mutka, K., & Jarvinen, H. (2005). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin, 37*(3), 14–18.
Lau, W. W. F., & Yuen, A. H. K. (2009). Exploring the effects of gender and learning styles on computer programming performance: implications for programming pedagogy. *British Journal of Educational Technology, 40*(4), 696–712.
Lawrence, R. (2004). Teaching data structures using competitive games. *IEEE Transactions on Education, 47*, 459–466.
Lee, F. L., & Heyworth, R. M. (2000). Problem complexity: a measure of problem difficulty in algebra by using computer. *Education Journal, 28*(1), 85–107.
Levy, Y. (2007). Comparing dropouts and persistence in e-learning courses. *Computers & Education, 48*(2), 185–204.
Lilley, M., Barker, T., & Britton, C. (2004). The development and evaluation of a software prototype for computer-adaptive testing. *Computers & Education, 43*(1), 109–123.
Manne, F. (2000). Competing in computing. In *Proceedings of the 2000* (pp. 129–138). Norsk Informatikkonferanse.
McDowell, C., Werner, L., Bullock, H., & Fernald, J. (2006). Pair programming improves student retention, confidence, and program quality. *Communications of the ACM, 49*(8), 90–95.
Mckinney, D., & Denton, L. F. (2006). Developing collaborative skills early in the CS curriculum in a laboratory environment. In *Proceedings of the 37th SIGCSE* (pp. 138–142).
Mory, E. H. (2004). Feedback research revisited. In D. H. Jonassen (Ed.), *Handbook of research on educational communications and technology* (pp. 745–784). Mahwah, NJ: Lawrence Erlbaum Associates.
Murphy, L., Richards, B., McCauley, R., Morrison, B. B., Westbrook, S., & Fossum, T. (2006). Women catch up: gender differences in learning programming concepts. *ACM SIGCSE Bulletin, 38*(1), 17–21.
Oncu, S., & Cakir, H. (2011). Research in online learning environments: priorities and methodologies. *Computers &Education, 57*(1), 1098–1108.
Pillay, N., & Jugoo, V. R. (2005). An investigation into student characteristics affecting novice programming performance. *SIGCSE Bulletin, 37*(4), 107–110.
Revilla, M. A., Manzoor, S., & Liu, R. (2008). Competitive learning in informatics: the UVa online judge experience. *Olympiads in Informatics, 2*, 131–148.
Regueras, L. M., Verdú, E., Muñoz, M. F., Pérez, M. A., de Castro, J. P., & Verdú, M. J. (2009). Effects of competitive e-learning tools on higher education students: a case study. *IEEE Transactions on Education, 52*(2), 279–285.
Regueras, L. M., Verdú, E., Pérez, M. A., De Castro, J. P., & Verdú, M. J. (2007). An applied project of ICT-based active learning for the new model of university education. *International Journal of Continuing Engineering Education and Life-Long Learning, 17*(6), 447–460.
Regueras, L. M., Verdú, E., Verdú, M. J., & de Castro, J. P. (2011). Design of a competitive and collaborative learning strategy in a communication networks course. *IEEE Transactions on Education, 54*(2), 302–307.
Rice, W.R., Beg, Y., Waters, C., Bokreta, M.K., & Santiago-Aviles, J.J. (1999). New pedagogical approaches to computer science education: a case study in peer learning. In *Proceedings of the 29th ASEE/IEEE Frontiers in Education Conference* (pp.13A4/12–13A4/15).
Roessling, G., & Hartte, S. (2008). WebTasks: online programming exercises made easy. In *Proceedings of the 13th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (pp. 363).
Rongas, T., Kaarna, A., & Kalviainen, H. (2004). Classification of computerized learning tools for introductory programming courses: learning approach. In *Proceedings of the IEEE international conference on advanced learning technologies* (pp. 678–680).
Siddiqui, A., Khan, M., & Katar, S. (2008). Supply chain simulator: a scenario-based educational tool to enhance student learning. *Computers &Education, 51*(1), 252–261.
Skiena, S. S., & Revilla, M. (2003). *Programming challenges*. New York: Springer-Verlag.
Spacco, J., Hovemeyer, D., Pugh, W., Emad, F., Hollingsworth, J. K., & Padua-Perez, N. (2006). Experiences with Marmoset: designing and using an advanced submission and testing system for programming courses. *ACM SIGCSE Bulletin, 38*, 13–17.
Straub, D. W. (1989). Validating instruments in MIS research. *MIS Quarterly, 13*(2), 147–169.
Tauer, M., & Harackiewicz, J. M. (2004). The effects of cooperation and competition on intrinsic motivation and performance. *Journal of Personality and Social Psychology, 86*(6), 849–861.
Tremblay, G., Guerin, F., Pons, A., & Salah, A. (2008). Oto, a generic and extensible tool for marking programming assignments. *Software – Practice and Experience, 38*, 307–333.
Trotman, A., & Handley, C. (2008). Programming contest strategy. *Computers & Education, 50*(3), 821–837.
Truong, N. (2007). *A web-based programming environment for novice programmers*. PhD thesis. Queensland: University of Technology.
Van der Pol, J., Van den Berg, B. A. M., Admiraal, W. F., & Simons, P. R. J. (2008). The nature, reception, and use of online peer feedback in higher education. *Computers & Education, 51*(4), 1804–1817.
Van Vugt, M., De Cremer, D., & Janssen, D. P. (2007). Gender differences in cooperation and competition - the male-warrior hypothesis. *Psychological Science, 18*(1), 19–23.
Vansteenkiste, M., & Deci, E. L. (2003). Competitively contingent rewards and intrinsic motivation: can losers remain motivated? *Motivation and Emotion, 27*(4), 273–299.

Vekiri, I., & Chronaki, A. (2008). Gender issues in technology use: perceived social support, computer self-efficacy and value beliefs, and computer use beyond school. *Computers & Education, 51*(3), 1392–1404.
Verdú, E., & Lorenzo, R. M. (2010). Effective strategies for learning: collaboration and competition. In *A new learning paradigm: Competition supported by technology* (pp. 11–40). Madrid: Sello Editorial.
Vihtonen, E., & Ageenko, E. (2002). Viope-computer supported environment for learning programming languages. In *Proceedings of the international symposium on technologies of information and communication in education for engineering and industry* (pp. 371–372).
Wang, F. L., & Wong, T. (2008). Designing programming exercises with computer assisted instruction. *Lecture Notes in Computer Science, 5169*, 283–293.
Wauters, K., Desmet, P., & Van den Noortgate, W. (2010). Adaptive item-based learning environments based on the item response theory: possibilities and challenges. *Journal of Computer Assisted Learning, 26*, 549–562.
Wiedenbeck, S., LaBelle, D., & Kain, V.N.R. (2004). Factors affecting course outcomes in introductory programming. In *Proceedings of the 16th workshop of the psychology of programming interest group* (pp. 97–110).
Yang, Y.-F. (2010). Students' reflection on online self-correction and peer review to improve writing. *Computers & Education, 55*(3), 1202–1210.