A Course on Algorithms and Data Structures Using On-line Judging

Ginés García-Mateos
Dept. de Informatica y Sistemas
Faculty of Computer Science
University of Murcia, Murcia, Spain
ginesgm@um.es

José Luis Fernández-Alemán
Dept. de Informatica y Sistemas
Faculty of Computer Science
University of Murcia, Murcia, Spain
aleman@um.es

ABSTRACT

High dropout rates are commonly the main problem we must face in Computer Science degrees. There are two main causes of dropout: the implicit complexity of the matter, and a lack of motivation among students. The second-year programming course of our university suffered from dropout rates over 70% of the more than three hundred enrolled students. In order to overcome this problem, we have adopted a new teaching methodology based on two key ideas: *replacing* the final exam with a series of activities in a continuous evaluation context; and making those activities more appealing to the students. In particular, most of the activities are designed as on-line programming competitions; they are carried out by using a web-based automatic evaluation system, the *on-line judge*. Experimental results show the high effectiveness of the proposed approach. On average, the dropout rate decreased to 40% while the pass rate doubled. Some strategies are used to ensure the authorship of the programs and to detect source code plagiarism.

Categories and Subject Descriptors

K.3.2 [**Computers and education**]: Computer and Information Science Education - computer science education

General Terms

Algorithms, Data Structures, Design, Experimentation, Human Factors, Theory

Keywords

Active learning, experience report, on-line judging

1. INTRODUCTION

Currently, educational tendencies are centered in the student's point of view rather than the instructor's one. A clear example of this trend is the *European Space of Higher Education*, whose purpose is to develop new teaching methodologies based on the students' learning process. The intention is to create independent, reflective and life-long learners. The new methods should stimulate students interest and offer appealing material, fair assessment and appropriate feedback.

Based on those principles, this paper describes an innovative experience with a course of "Algorithms and Data Structures" (ADS), using a web-based automatic judging system called Mooshak [8]. The goal of the study is to analyze the effect of this novel teaching methodology in the dropout and pass rates. Our preliminary results prove the viability of the proposal, and a high capacity to generate motivation and enthusiasm among students.

The rest of the paper is organized as follows. Section 2 presents a brief review of some related work. Then, we introduce and justify in Section 3 the methodological approach of the proposal. Section 4 offers the main results of the learning experience applied to 337 students in a second-year course for computing majors on ADS. In Section 5, we analyze and discuss the results achieved by employing this new methodology. The last section presents some concluding remarks and outlines the efforts to be made in the future.

2. RELATED WORK

In the literature, most authors reach the same conclusion: learning to program is difficult [9]. For example, some studies point out that it takes approximately ten years to transform a novice into an expert programmer [11]. A large number of techniques and methods have been proposed to improve students comprehension in computer programming courses [9], both in CS1 and CS2. E-learning methods constitute a viable and promising alternative in programming pedagogy. Particularly, on-line judging systems have already been applied in this discipline.

Guerreiro and Georgouli [7, 6] propose an e-learning educational strategy in first-year programming courses. They adopt Mooshak automatic judging system for grading lab assignments and for self-assessment purposes. Automatic evaluation accounts for about 30% of the final mark. This approach provides important benefits in a CS1 course. A well thought out set of test cases prevents wrong programs sent by students from passing test runs. As a consequence, students must be much more rigorous in developing their programs. Likewise, students obtain immediate feedback from Mooshak. Another advantage of their proposal is the objectivity of the evaluation. Moreover, the authors consider that teachers can save time and work if an automatic judging system is used. Nevertheless, important concepts such as robustness and legibility need to be manually graded by the instructors.

ITiCSE '2009 Paris, FRANCE

In order to address this issue, Bowring [2] proposes a new paradigm for programming competitions where the quality rather than the fast completion of the programs is evaluated. Both technical and artistic merit are taken into account as judging criteria. According to the author, technical quality refers to how well submissions meet the stated requirements, whereas artistic quality is related to the organization of the code, the readability of the code and its documentation, and the readability of other artifacts such as output files.

Our novel contribution resides in the way of using the on-line judging system. We take Guerreiro and Georgouli's strategy one step further, by completely replacing the traditional "final exam evaluation" with a series of activities, most of them using Mooshak. Thus, two important benefits are obtained: (i) students are very motivated to take part in the proposed activities, with the hope of avoiding the final exam; and (ii) the work of the students is evaluated along the course, rather than just in a single final exam.

3. METHODOLOGICAL APPROACH

In this section we describe in detail the proposed approach. Firstly, we identify the pedagogical principles that have guided our efforts. Then, the basic aspects of on-line judging systems are presented. Finally, we analyze how to introduce this tool into the learning process.

3.1 Pedagogical Principles

Before our pioneering experience, the main problems observed in the course on ADS were a low motivation and participation of the students in class, that resulted in very high dropout rates. In previous years, approximately between 2/3 and 3/4 of the more than 300 students enrolled in the course dropped out. With the aim of reversing this trend, we decided to adopt a new learning paradigm. The four pedagogical principles underpinning this novel paradigm, following some recommendations by other authors [10], are:

- **Motivation.** With dropout rates around 70%, increasing the motivation of students is clearly essential. By motivation we mean the desire to learn new concepts and methods, and to put them into practice.
- **Active learning.** The students have to be involved in, and conscious of, their own learning process. An active methodology, where students are not mere spectators, is necessary to achieve real and long-lasting learning.
- **Autonomous work.** We believe that the best way to learn computer programming is by programming. A simple memorization of concepts is nearly useless. The students have to reflect on algorithmic problems by themselves and autonomously create their programs.
- **Feedback of the learning process**, considered from the point of view of the students. Traditionally, they just obtain a final mark at the end of the course. The method should provide students with a continuous evaluation on how well they are doing.

All of these objectives require more participatory activities, both in class and out of class. In these situations is where e-learning tools can produce a great benefit, not by substituting the teachers, but by helping them to control and evaluate the activities.

3.2 On-line Judging

One of the key elements of our approach is the on-line judging system. This system is an automatic tool which is able to evaluate the correctness of computer programs, based on a predefined set of pairs input/output. It has a web-based interface, which is different for the students, teachers, guest users and the system administrator. More specifically, we are using Mooshak 1.4 [8], which is free and publicly available. This system was originally created to manage programming competitions. However, Mooshak is applied more and more to computer programming learning.

Compared to other disciplines, judging the correctness of a program, with a high degree of certainty, is relatively simple; that is what makes automatic evaluation feasible. However, there are many aspects of programming that are not so easy to evaluate: computational complexity, design and organization of the code, robustness, legibility, etc. In consequence, the task of human instructors remains to be essential.

3.3 A Judge-based Methodology

By upholding the pedagogical principles proposed in subsection 3.1, we have designed four different ways of using the on-line judge in the course:

- **Independent problems.** In this kind of activity, many problems are proposed to the students. The problems are independent of each other, and with different levels of difficulty. The students are expected to select and solve some of them, not necessarily all. Problems can be grouped by category, in such a way that each category illustrates a programming technique discussed in class. Typically, some weeks are given to complete the activity.
- **Dependent problems.** This case is preferable when the objective of an activity is to develop a longer and more complex programming project. The project is divided into smaller and consecutive subproblems; each of them is described as a problem in the judge. The students have to solve all the problems in the given order. In the itinerary, a number of programming techniques can be illustrated. This kind of activities can normally take several months.
- **Contest-style.** Contrary to the other cases, where activities are carried out on-line, here the presence of the students is required. A set of at most 9 problems is given to the participants. They have to try to solve as many problems as possible, and as fast as they can. The contest can take between 4 and 6 hours.
- **Designing problems.** This is the most creative type of activities. The students have to create a problem with the format of the judge: problem description, source code to solve it, input cases, and expected outputs. The ability of the students to produce original and relevant problems is evaluated here.

These activities are closely related to the six cognitive levels of Bloom's taxonomy [1]. Each type of activity is specially focused in a particular cognitive level. For example, the first type is centered in *application* abilities; the second type tries to improve *synthesis* skills; the third type aims at developing *knowledge* and *evaluation*; and the last type is focused on *comprehension* and *analysis*.

4. EVALUATION OF THE METHOD

In this section, we provide detailed information about the groups where the experiment was carried out, the application of the methodology, and the obtained results. The study was conducted at the Computer Science Faculty of the University of Murcia (Spain).

4.1 Participants and Background

Table 1 summarizes the main information regarding the course and degrees under study. Observe that the course on ADS is present in three different degrees. Two of these groups (TECS and TECM) can be considered as highly populous, while the other (CSE) is a reduced group.

Table 1: Course and degrees where the new method was applied. "ECTS": equivalent load in European Credit Transfer System [5]; "Enr.07": enrollment the year before the experience; "Enr.08": enrollment the year of the experience. CSE is a 5-years degree, while TECS and TECM are 3-years degrees.

Course name	Year	Duration	ECTS
Algorithms and data structures (ADS)	2nd year	Annual	12
Degrees	**Acronym**	**Enr.07**	**Enr.08**
Computer science engineering	CSE	56	44
Technical engineering in computer systems	TECS	162	162
Technical engineering in computer management	TECM	150	131

ADS is basically an advanced course on programming, emphasizing issues of algorithms and data representation. This course introduces such topics as data structures, abstract data types and formal specifications. The course also includes techniques for performance analysis and design of algorithms. The programming languages used to illustrate these concepts are C, C++ and Maude [4]. ADS was traditionally organized in a *monolithic* form: weekly lectures, laboratory sessions, final exam and a programming project for each semester.

4.2 Instantiation of the Method

In the academic year spanning from fall of 2007 to spring of 2008, the three groups were involved in the new methodology. The students were given the possibility to follow the traditional method, instead of the new one; however, more than 2/3 of them participated in the proposed activities. Each activity of the course is related to a unit of theory or practice. The students who passed all the activities did not have to do the final exam. The students who failed some activities of theory were given the possibility to pass those units by doing the corresponding part in the final exam.

Table 2 lists all the activities of the course. Observe the great variability of types: programming activities in Mooshak (individually and in groups), partial exams, and an experimental study. Some aspects are worth mentioning:

- In order to follow the new method –which we called *continuous evaluation* methodology–, students are required to attend a minimum of 80% of the lectures. This way, we emphasize the importance of studying and learning along the course.

Table 2: Description of the activities of the course: corresponding to theory (U) and practice (P) units. The activities that use Mooshak are marked in gray. "Type": independent problems (I), dependent problems (D), contest-style (C), partial exam (E), experimental efficiency study (S). "#p.": number of problems in the judge. "Mode": activities that should be done individually, or in groups of two. The last two activities are optional, the rest are compulsory.

Activity	Type	#p.	Language	Mode
U1. Formal specifications	I	26	Maude	Group
U2. Hash tables exam	E	-	-	Individual
U3. Trees exam	E	-	-	Individual
U4. Graphs	I	15	C/C++	Individual
U5. Algorithmic analysis exam	E	-	-	Individual
P1. Data structure implementation	D	17	C++	Group
P2. Experimental analysis	S	-	-	Group
U6. Divide and conquer	D	16	C/C++	Individual
U7. Greedy algorithms	I	7	C/C++	Individual
U8. Dynamic programming	I	7	C/C++	Individual
U9. Backtracking	I	12	C/C++	Individual
U10. Branch and bound	I	12	C/C++	Individual
Local programming contest	C	8	Java/C/C++	Group

- All the activities have to be documented by the students and handed over within the prescribed deadline. Then, the documents are corrected by the teachers and feedback is given to the students right away.

- When introducing the on-line judge in the course, most work is not done in the presence of the instructors. One of our main concerns was to guarantee the originality and authorship of the submitted programs. For that reason, we use a plagiarism detection system developed by Cebrian et al. [3]. Thanks to Mooshak, all submissions are available in the judge's server, so the plagiarism detector can be easily applied. Also, some activities include a compulsory interview with the students, in order to demonstrate their authorship.

4.3 Results of the On-line Judge

Overall, the impact of on-line judging in the teaching of ADS has been dramatic and very positive. Up to 273 of the 337 enrolled students (81%) participated in some activity of the judge; 268 of them (79.5%) solved at least one problem. In total, the on-line judge received 16054 submissions. This makes an average of 59 submissions per student: 44 C/C++ programs, and 15 Maude programs. The on-line judge classified 6427 submissions as correct (40.1%). Each student attempted an average of 20 problems and managed to solved 18.5. More information on the classification of these submissions is shown in Table 3 and Figure 1.

The average number of submissions per student until getting a program accepted is 2.7; however, many students found the solution to the problems at their first attempt (in fact, mode is 1). There is also a clear difference in the three groups under study. For example, while the average acceptance rate in CSE is 50%, in TECS it is 37%, and in TECM 36%; this difference is consistently observed in all the proposed activities. In part, this can be explained by the different nature of the groups; recall that CSE is a 5-years degree with higher entry requirements than the 3-years degrees TECS and TECM.

Table 3: Detail of the classification of the submissions by activity, as listed in Table 2. The last column indicates the number of students (or groups of two) that passed the corresponding activity.

Activity	Total submissions	Correct	Wrong answer	Runtime error	Other errors	Pass the activity
U1	4085 (25%)	1971 (48%)	1511 (37%)	600 (15%)	3 (0.1%)	85 groups
U4	2884 (18%)	1164 (40%)	493 (17%)	366 (13%)	861 (30%)	181 (54%)
P1	3615 (22%)	1229 (34%)	1099 (30%)	273 (8%)	1014 (28%)	41 groups
U6	2032 (12%)	705 (35%)	263 (13%)	161 (8%)	903 (44%)	178 (53%)
U7	1780 (11%)	619 (35%)	688 (39%)	66 (4%)	407 (23%)	182 (54%)
U8	830 (5%)	410 (49%)	189 (23%)	51 (6%)	180 (22%)	170 (50%)
U9	778 (4%)	309 (40%)	171 (22%)	12 (2%)	286 (37%)	162 (48%)
U10	50 (0.3%)	20 (40%)	3 (6%)	3 (6%)	24 (28%)	13 (4%)
Total	16054	6427 (40%)	4417 (28%)	1532 (10%)	3678 (23%)	75 (22%)

Figure 2 shows a histogram of the number of problems solved by each student. This value covers quite a large range, from 1 to 80, with an average of 18.5, standard deviation of 14.7, and with two modes of 8 and 12. The minimum number of problems necessary to pass all the activities was 20.

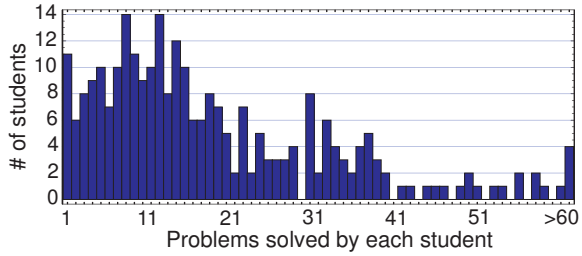

Figure 1: Number of problems solved by each student in Mooshak.

5. ANALYSIS AND DISCUSSION

The results obtained after the application of the judge-based methodology are very promising. Considering just the final marks of the students, shown in Table 4, the new approach achieved excellent improvements. We can see a significant increase in the pass rate, from 10.86% to 22.25%. However, the most striking fact is the dramatic decline in the dropout rate, from 72.28% to 44.80%. The new methodology encourages students to make them get back on pace again. The increase in the failure rate is due to the high number of students that pass some activities (normally above 50% in each activity) but not all. However, we think they are better prepared to pass the course in subsequent attempts.

Table 4: Pass, failure and dropout rates of the three degrees where the new methodology was applied (in 2008), and the results in the previous year (2007).

Final results	Total		CSE		TECS		TECM	
	2007	2008	2007	2008	2007	2008	2007	2008
Pass rate	11% (40)	22% (75)	20% (11)	61% (27)	7% (11)	13% (17)	11% (18)	19% (31)
Failure rate	17% (62)	33% (111)	14% (8)	7% (3)	23% (34)	32% (42)	12% (20)	41% (66)
Dropout rate	72% (266)	45% (151)	66% (37)	31% (14)	70% (105)	55% (72)	77% (124)	40% (65)

5.1 Quantitative Analysis of the Results

Here we will do a comparative analysis of the final marks in the ADS course; all students in the year 2007 are considered as the control group, and those in 2008 are the experimental group. It is important to note that dropout students are not considered here.

Figure 2 shows the grouped marks (on a scale of 0-10) as two overlapping histograms. The scores for the control group have a mean of 4.59, standard deviation of 2.38, and a mode of 4. The mean, standard deviation and mode for the experimental group were 5.02, 2.23 and 4, respectively. With the usual 95% confidence interval, a t-test for independent samples determined these results to be statistically significant, $t(283)=1.36$, $p<0.01$.

As a consequence, the proposed method not only reduces the number of dropout students, but also has an important benefit in the scores of the rest of students.

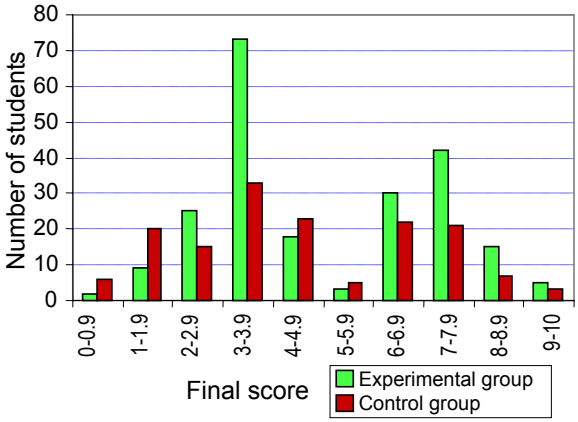

Figure 2: Final marks of the students in ADS, for the control group and the experimental group.

On the other hand, the mean number of attempts to pass the course was 1.67 for the control group, and 1.29 for the experimental group. The relationship between the number of attempts and the final marks was investigated using Pearson correlation analysis. No significant correlation (-0.06) was found for the experimental group, whereas results of the study showed that there is a low positive correlation (0.30) for the control group. This means that many students are able to pass the course at their first attempt, while in the past more than one attempt was normally required.

5.2 Qualitative Analysis

Apart from the previous results, there is evidence that the proposed organization of the course successfully meets all the pedagogical principles established in subsection 3.1:

- **Motivation.** In all the ways of using the on-line judge, the ranking plays a fundamental role in motivating students to solve more problems, faster and more efficiently. This is evident in the high rate of students –an average of 7%– who solved more problems than those necessary to obtain the maximum mark: they also wanted to be up in the ranking. On the other hand, since submissions are public, students do not have to feel frustrated when they receive a "wrong answer". They can watch that their classmates are going through the same troubles, and it is a part of learning.

- **Active learning.** When students solve the proposed problems, they play a leading role in their own learning. For example, it is worth underlining that 1119 questions were asked by the students in Mooshak; usually they were answered by the teachers within a day.

- **Autonomous work.** Students can work in the labs of the Faculty, where they have help from the teachers. However, it is evident that students work most of the time at home, and ask questions to the teachers by using Mooshak. Submissions done by the students outside laboratory time represent a total of 87.5%, thus demonstrating the importance of autonomous work.

- **Feedback of the learning process.** The web system provides feedback to help students to correct many errors in their programs, thus avoiding assistants spending much effort figuring out the causes of the failure, as it happens in a traditional evaluation. The judge is accessible 24-hours a day and the feedback is instantaneous. From the point of view of the teachers, information is also comprehensive and immediate; they can analyze the difficulty of the problems, the evolution of the students, identify the best students, etc.

To conclude, we can mention some remarkable results from a survey carried out among the students, regarding their experience with the new methodology. They were asked to indicate a degree of agreement/disagreement with a series of statements. These are some of the items evaluated:

- 77% of the students agree that "they learn better with the new methodology than with the old one";

- 68% of them say that "the public ranking of the judge fosters competitiveness";

- 77% agree that "the judge has been a very useful tool in all the activities"; all students disagree with the statement "cheating is easier with the on-line judge";

- 91% say that "if they could choose, they would follow again the continuous evaluation methodology".

6. CONCLUSIONS

We have described in this paper an innovative experience on computer science education. In general, the results of our experiment are excellent. The proposed combined strategy (continuous evaluation/on-line judging) has achieved its primary goal of reducing the dropout rate, from 72% to 40%. As a consequence, the pass rate increased, and so did the failure rate. On average, the final marks of the students are higher with the new methodology; however, the high failure rate remains a challenge to be faced in the future.

As we have discussed, most of the benefits are due to the use of on-line judging. This approach improves self-assessment skills and encourages students to work independently. The public ranking and other data provided by Mooshak promote competitiveness. The assessment is fair and objective, and students are able to gain additional feedback from the human judges. Students get themselves more involved into their learning, thus contributing to build a strong foundation for the student's life-long learning.

We believe that the proposed methodology can be easily applied to other courses in a computer science degree, particularly in those which include programming assignments. It could also be an interesting direction to investigate its application to computing students with disabilities.

Two major aspects remain to be improved in the future: the feedback provided by the judge, and plagiarism detection. We are currently working on extensions of Mooshak to provide detailed feedback in case of "wrong answer", and to integrate the plagiarism detection subsystem into the web interface.

7. REFERENCES

[1] Bloom, B. S.: Taxonomy of Educational Objectives: The Classification of Educational Goals. Handbook I, Cognitive Domain. New York: David McKay (1956)

[2] Bowring, J. F.: A New Paradigm for Programming Competitions. SIGCSE Bull. 40 (1): 87–91 (2008)

[3] Cebrian, M., Alfonseca, M., Ortega, A.: AC: An Integrated Source Code Plagiarism Detection Environment. In: CoRR abs/cs/0703136 (2007)

[4] Clavel, M., Duran, F., Eker, S., Lincoln, P., Marti-Oliet, N., Meseguer, J., Talcott, C.: All About Maude - A High-Performance Logical Framework. LNCS Vol. 4350 (2007)

[5] ECTS - European Credit Transfer and Accumulation System, European Commission 2003. `http://ec.europa.eu/education/programmes/socrates/ects/index_en.html`

[6] Guerreiro, P., Georgouli, K.: Enhancing Elementary Programming Courses Using E-learning with a Competitive Attitude. International Journal of Internet Education (2008)

[7] Guerreiro, P., Georgouli, K.: Combating Anonymousness in Populous CS1 and CS2 Courses. In: Proc. ITICSE 2006, 8–12 (2006)

[8] Leal, J. P., Silva, F. M. A.: Mooshak: a Web-based, Multi-site, Programming Contest System. Software-Practice, and Exper., 33(6) 567–581 (2003)

[9] Robins, A., Rountree, J., Rountree, N.: Learning and Teaching Programming: A Review and Discussion. Computer Science Education, 13(2) 137–172 (2003)

[10] Vrasidas, C.: Issues of pedagogy and design in e-learning systems. ACM symposium on Applied computing, 911–915 (2004)

[11] Winslow, L. E.: Programming Pedagogy – a Psychological Overview. SIGCSE, 28: 17–22 (1996)