International Journal of Engineering Education Vol. 27, No. 4, pp. 867–874, 2011
Printed in Great Britain

0949-149X/91 $3.00+0.00

Assessment Technique to Encourage Cooperative Learning in a Computer Programming Course*

MIGUEL AREVALILLO-HERRÁEZ and JOSÉ M. CLAVER
Computing Department, University of Valencia, Spain, Avda. Vicente Andrés Estellés s/n, 46100 Burjassot, Spain.
E-mail: {miguel.arevalillo,jclaver}@uv.es

Cooperative learning has been reported to produce greater student achievement than other traditional learning methodologies. However, difficulties are usually found with plagiarism and with achieving the equal commitment of all members. In this paper, we propose a teaching methodology that aims at avoiding plagiarism, promoting cooperation and encouraging participation of group members. To this end, a number of collaborative tasks are set using a typical Problem Based Learning (PBL) approach. Instead of evaluating student performance by means of a typical report (or portfolio), students are assessed by means of a written test, composed of questions that are closely related to the collaborative task. Furthermore, student grades are made dependent on the individual grades obtained by all other members in the group. In this way, students are encouraged to help each other, and their level of commitment is increased. This technique has been applied to a group of 46 students taking the 'Programming Languages' module, a third level module of the Degree in Computing at the University of Valencia (Spain). Results have shown a significant improvement in student performance. Moreover, individual questionnaires have provided evidence that students prefer this technique to other more conventional teaching methods.

Keywords: collaborative learning; active learning; problem based learning; plagiarism

1. Introduction

Active learning is generally defined as any instructional method that engages students in meaningful learning activities [1]. In this context, active learning activities refer to those that involve students in doing things, and thinking about what they are doing. A typical active learning strategy is Problem-Based Learning (PBL), an instructional method where relevant problems are introduced at the beginning of the instruction cycle, and used to provide the context and motivation for the learning that follows [2]. Although it is not necessary, PBL is often combined with cooperative learning, a structured form of group work where students pursue common goals while being assessed individually [3].

Despite the benefits of using cooperative strategies in computer assignments, these need to be carefully designed so that collaboration is integrated into assignments and assessments [4]. In addition, the following five components should be included: individual accountability, mutual positive interdependence, face-to-face promotive interaction, appropriate practice of interpersonal skills, and regular self-assessment of team functioning [5]. Apart from this, specific mechanisms are required to deal with plagiarism and the presence of uncooperative members that hinder the work of the rest of the team. In this paper, we present an integrated teaching and assessment method applied in a computer programming course module, one which inherently contributes to avoiding both of these problems.

Previous proposals in this direction have mainly focused on evaluation strategies that consider the individual's contribution to group projects as a means of enhancing the participation of group members in cooperative work, for example, [6, 7]. Relative evaluation has also been common, despite the negative effects on student interaction, which can become an obstacle to cooperative learning [8].

Conferences and books devoted entirely to plagiarism denote an increasing concern about this topic, and two major schools of thought can easily be identified. The first is based on detection, while the other relies on prevention.

On the detection side, research has mainly focus on the design, provision and usage of Plagiarism Detection Software/Services (PDS). This may focus on detected cut and pasted text from known sources of information (including the Internet) [9]; or be designed to detect more complex forms of plagiarism (such as programming code). Examples in this last category are [10, 11]. A review of existing PDS can be found in, for example, [12], and a very recent investigation about the limitations of current PDS is provided in [13], including proposals for using various existing technologies to tackle them in future systems.

On the prevention side, two major approaches have been adopted. On the one hand, a period of academic apprenticeship has been proposed as a

* Accepted 10 March 2011.

means of decreasing plagiarism. For example, the use of an anti-plagiarism tutorial has been shown to reduce the likelihood of plagiarism [14] substantially, and Bolin [15] proposes spending more time and effort on teaching students accepted definitions of plagiarism and how to work within the conventions of academic writing before resorting to detection when suspicious writing warrants it. On the other hand, some authors (for example [16]) defend the use of course design and assessment as instruments to deter plagiarism. Setting assignments that require that students think critically and make use of their reasoning skills is one such strategy that has been defended by several authors (for example [17, 18]).

The technique presented in this paper can be classified in this last group of strategies. However, instead of asking students to think critically, the evaluation of the collaborative tasks has been replaced by a written test composed of questions that are closely related to the work performed. Then, individual grades take into consideration the grades obtained by all other members who were involved in the collaborative task. Although the strategy is presented in the context of a third year module of a Computing degree, it can easily be extended and applied in courses in other levels and degrees, as long as the collaborative tasks can be evaluated by a written examination.

2. The module

'Programming Languages' is a compulsory course module delivered at the third level of the degree in Computing at the University of Valencia (Spain). This is a two-semester course that presents some of the major programming paradigms. The module is composed of a total of 90 contact hours, which are delivered over 30 weeks (at a rate of 15 weeks per semester). From week 1, students receive a two-hour lecture session per week. Laboratory sessions last for 3 hours and take place once every two weeks, starting at week 3 of each semester.

During the first semester, the subject focuses on object-oriented programming, generic programming, event-based programming and concurrent programming; Java programming language is used. The second semester deals with logic and functional programming, and concepts are illustrated using PROLOG and LISP, respectively.

The objectives of this module are not limited to teaching programming skills. On the contrary, a number of important concepts that require a student to change his or her way of thinking are taught. In this respect, this module is especially suitable for the implementation of collaborative learning strategies. In general, programming solutions are not unique, and this type of environment allows students to explore alternative problem solutions through discussion. This helps to clarify ideas, and contributes to the development of critical thinking skills. As an additional benefit, this style of learning benefits the improvement of other more general competencies, such as team work, social interaction and communication skills.

3. The teaching method

In previous years, major efforts have been made to incorporate PBL and cooperative learning strategies into the course. Unfortunately, many of the attempts have not achieved the success expected, mainly because of plagiarism and the presence of shirkers. During the academic year 2008–9, a teaching method that placed special emphasis on these two issues was designed and successfully applied.

In this section, the teaching strategy is described in detail. To allow for a comprehensive evaluation of the method, it was applied only during the first semester. In this way, comparative analyses both between the two semesters, and with the previous academic year, were made possible.

The teaching method consists of a series of iterations of the loop shown in Fig. 1. The contents delivered in the first semester have been divided into four learning units, namely: principles of object oriented programming; UML notation; exceptions and event-based programming; and concurrent programming. Each of these units is first introduced during the lecture sessions, and the main concepts

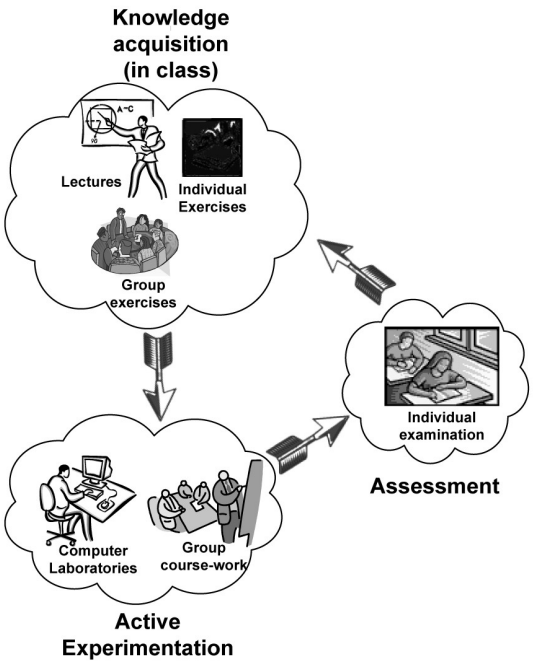


Fig. 1. Iterative teaching method.

are practised through a number of in-class individual and group-based exercises. Then students are required to work cooperatively on some further activities aimed at gaining a more in-depth understanding of the learning unit. To close the loop, the student knowledge on the unit is evaluated individually. In order to encourage cooperation, the grade of a student in a learning unit depends on the grades obtained by all the other members of the same group. Further details about each loop stage are provided below.

3.1 Knowledge acquisition

This task is mainly related to the first three levels of Bloom's taxonomy (knowledge, comprehension and application) [19]. A series of lectures and individual and group exercises are performed at the first stage of the learning loop. These aim at providing an understanding of the basics of each learning unit. To achieve a clear alignment between learning objectives and the last stage of the loop (assessment), the first lecture activity consists of stating the learning objectives for the unit, in a form that facilitates self-assessment by reflective thinking [20]. Then, lectures are combined with in-class exercises that students have to hand in. Although these exercises are not used for grading purposes, they are corrected and used as a form of formative assessment to aid learning. Their purpose is two-fold: to provide feedback on the student's work, and to allow the lecturer to monitor student learning better.

As an example, in the UML learning unit, the basics of the notation are introduced in class, interspersing the explanations with some simple exercises that students have to resolve in teams and hand back to the lecturer. To avoid the effort of correcting these exercises on an individual basis, these are resolved in class (with the active participation of the students, for example stimulating discussion or taking a student to the board). As such, the solutions handed in by students are used to detect possible misunderstandings and as a non-intrusive method of attendance control.

3.2 Active experimentation

Programming is a skill that requires practice, and thus this should have a major role in the delivery of the module. Once the foundations of the learning unit have been acquired, an active experimentation phase takes place.

This stage is composed of two main activities in which students work in teams of three to practise the concepts that they have learned during the previous stage. Both activities aim at covering the last two levels of Bloom's taxonomy (synthesis and evaluation) and promoting self-learning.

In a first activity, an exercise bulletin is handed to the students. These have been carefully structured in two parts and are designed to help knowledge building. Part A covers the fourth level of Bloom's taxonomy (analysis). Students are asked, for example, to examine pieces of code and predict the output. To encourage discussion and critical thinking, it is suggested that students do this task individually, discuss the results within the group and then execute the code on the machine to verify the results. In Part B, a series of programming tasks that require some further research on concepts not seen in class are requested. These are usually designed so that they admit multiple solutions, again in order to encourage cooperation and critical and creative thinking. To establish a clear relation between the activities and the learning objectives initially established for the unit, each exercise/task in the bulletin is explicitly related to one or more learning objectives.

The second activity comprises the laboratory sessions, for which a problem-based learning approach has been adopted. PBL is defined as a process of teaching that uses concrete problems to motivate students, and that focuses on student centred activities [21]. It has been widely used in the design of computing curricula (see, for example, [22]). In this case, students are given a problem that they have to analyse outside the class, commonly using some of the research concepts that they have worked on in Part B of the exercises bulletin.

In the case of the UML learning unit, Part A of the bulletin asks the student to produce skeleton codes from given UML diagrams and vice versa. The objective of these exercises is to establish a clear relation between the contents learned in this and the previous learning unit (principles of object oriented programming). Part B contains two types of exercises. In the first exercises, students are given pairs of UML diagrams and problem specifications, and they are requested to reflect a number of specification changes in the corresponding UML diagrams. The second type of exercises consists of producing UML diagrams for relatively simple specifications.

The laboratory session consists of a single more complicated exercise. A more complex specification is given, and students are requested first to produce the corresponding UML diagrams, and then to code the specification in Java, according to the UML produced. In all the exercises, the solutions to the problems are not unique, and students should be able to evaluate the most appropriate one. This encourages discussion and high-level thinking skills.

3.3 Assessment

Simply working in groups does not imply a form of cooperative learning. For cooperative learning to exist, it is necessary that students work together to

accomplish shared goals and maximize everyone's learning [23].

In the design of the assessment strategy for this course, a continuous evaluation approach has been adopted, and a special emphasis has been placed on facilitating self-assessment and encouraging cooperative learning.

Regarding self assessment, the objectives that are specified for each learning unit can easily be used as a checklist. In addition, once students have completed the laboratory sessions, a self-assessment working sheet with tentative exam questions (to be solved either individually or in teams) is handed to them, along with sample answers. This document helps the students to understand and recognize the desired standards, and it constitutes another form of formative assessment.

To encourage students to help each other and set a common goal, the final grade is composed of an individual and a team component. To close the learning loop, students have to take a final individual test for each learning unit. This test aims to measure the student's individual progress with reference to the stated learning objectives; it employs a format that is consistent with the course activities performed. Students have to take four such tests during the semester (one per learning unit), and the final grade is obtained from the average of the scores achieved at each one of them. However, each of these scores is influenced by the geometric average of those obtained by each member of the team. In particular, the individual score for each examination is averaged with the geometric average of the grades obtained by all team members. With this scheme, low grades strongly penalize the rest of the team members, and individual accountability is reinforced.

Although this may seem too strict, exam questions are set so that they are closely related to the analysis, synthesis and evaluation activities performed during the previous stages. In general, most questions in the examination are variations of some that appeared in the bulletins (in the case of UML, exercises that appeared in the bulletin, with sufficient modifications so that memorizing the answer is not sufficient to resolve the questions adequately). This makes it extremely rare that a student who has performed his or her work following the suggested guidelines obtains a low grade. At the same time, students who have not participated in the research work involved in the group assignments are likely to fail at these questions. This encourages post-discussion sessions to facilitate an explanation of the results among team members, which are useful even though some teams may decide to divide the work between their members. This favours learning by explaining, an approach for which there is solid evidence that it is a powerful learning method, leading to a deeper understanding when learning new material [24, 25].

In general, this assessment strategy establishes a common goal for the team, and enables student collaboration. On the one hand, weak students become morally obliged to make an effort, so that the grades of other team members are not negatively affected. On the other hand, they feel supported by the rest of the team members who are also interested in pulling their own grades up. Moreover, it forces teams to adopt a stronger position against shirkers. In many cases, members of a team might have to do an extra amount of work to compensate for the presence of a shirker. However, this assessment strategy causes shirkers to be rejected by other team members. To encourage this attitude, students were informed that conflicts and non-contributing members should be reported before the test date, otherwise the entire team would be obliged to accept the consequences of the impact on his or her low grade.

To increase the student's commitment to the group, they are allowed to group themselves as desired. Although most collaborative work researchers advocate random assignment, for example, [23], in this case an existing social relationship may constitute an extrinsic motivation that works in favour of the strategy.

3.4 Planning

A more detailed sequence of the teaching and learning activities involved at each learning unit is illustrated in Fig. 2. In this figure, each activity is also related to the most active levels of thinking in Bloom's taxonomy.

To be able to coordinate the activities in the loops, they have been organized in such a way that the collaborative work and the out-of-the-class activities for a learning unit are performed at the same time as the in-class activities that correspond to the next learning unit.

4. Results

In this section, two types of results obtained with the methodology presented in this paper are discussed, both referring to a group of 46 students taking the module described in Section 2.

First, the students' grades are compared with those obtained by the same students in the second semester (without using cooperative learning strategies) and to those obtained by the previous year's students in the same semester. Second, the results of a questionnaire filled out by students are also presented.

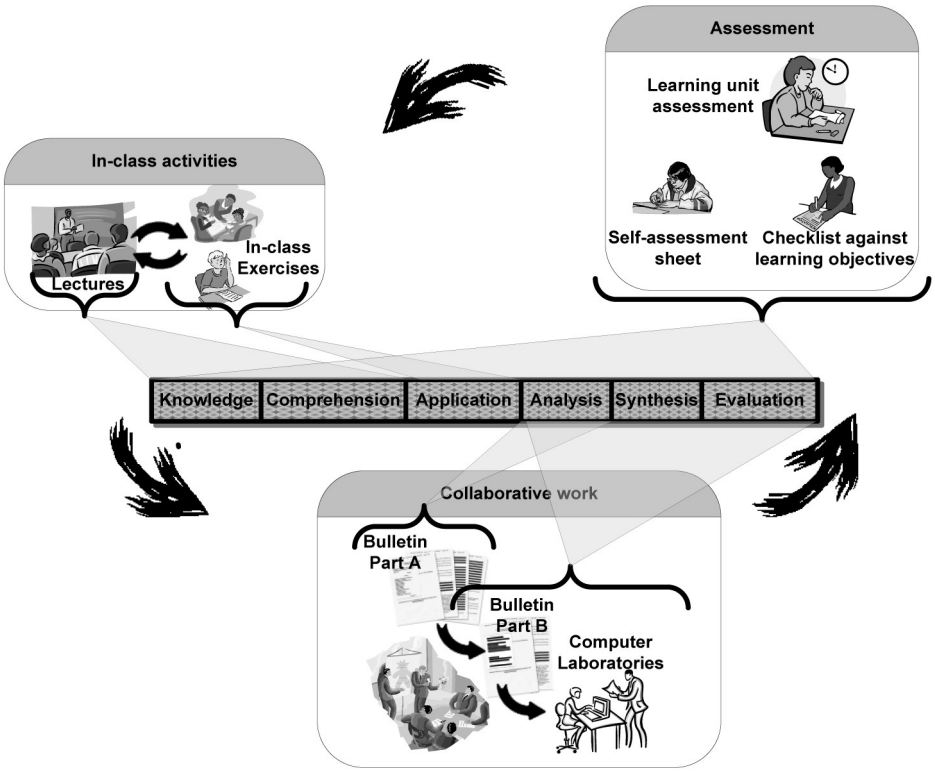


Fig. 2. Learning loop and relation with levels of thinking in Bloom's taxonomy.

4.1 Student grades

An ideal comparison of different learning methods would involve a study of the results obtained by the same students on the same contents using each of the methods. However, the inherent difficulties of this approach make it impossible to implement in practice.

Instead we have used this learning method during the first semester and compared the results obtained by the students against the following:

1. The results obtained by students attending the course module during the previous year. In this case, the contents are the same but the students are different (except for students re-taking the course).
2. The results obtained by the same students during the second semester. In this case, the students are the same but the contents are different.

Figure 3(a) shows the student dropout rates at each semester. It can be seen that the use of the

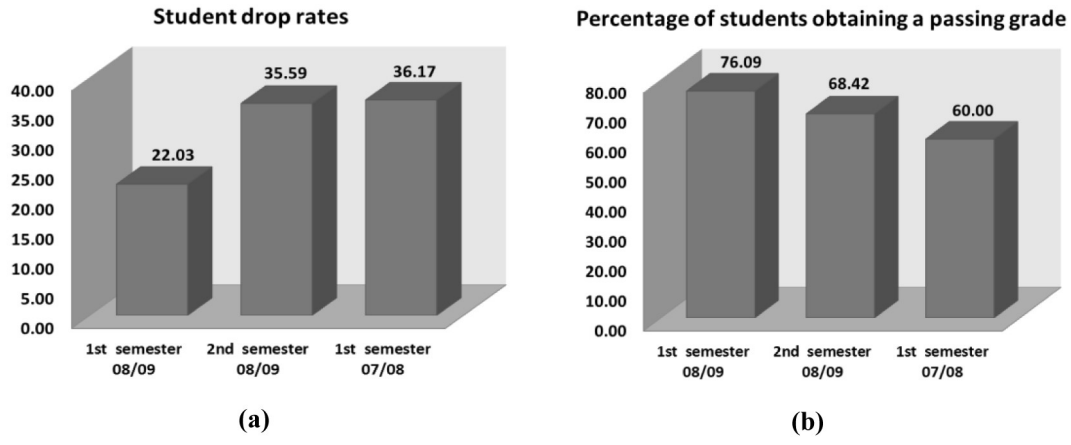


Fig. 3. (a) Student dropout rates as a percentage of the students enrolled at each semester. (b) Percentage of students who completed the module successfully (before re-sit examinations).

integrated learning strategy presented has had a large impact on this variable, leading to a significant reduction. Figure 3(b) also shows that the number of students who successfully completed the module has also increased (student dropouts have been excluded in the calculation of the percentages).

Another important result is related to the final grades obtained by the same students during the first and second semesters. In Fig. 4, a dispersion graph of these grades is presented to study possible correlations visually (these are specified according to the Spanish educational system, on a scale of 0 to 10, 5 being the minimum pass grade). Only grades of students who did not drop any of the semesters have been plotted. Surprisingly, not much correlation is observed. Some students who obtained a good grade during the first semester did not perform that well during the second (and vice versa). The bottom-right quadrant of this figure is especially telling in this respect, illustrating a significant number of students who passed the first semester but struggled during the second semester of the course. This effect may in part be produced by the students' preferred learning style, supporting existing theories that assert that this has a significant effect in their performance [26, 27].

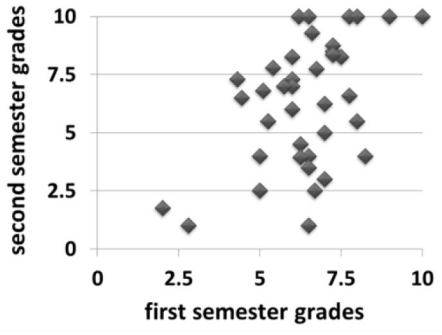
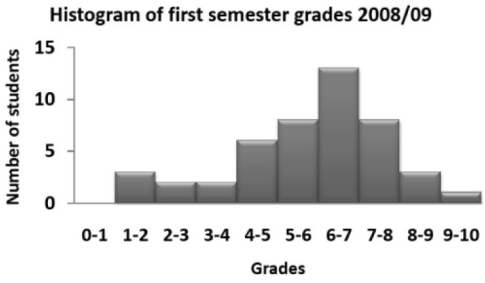
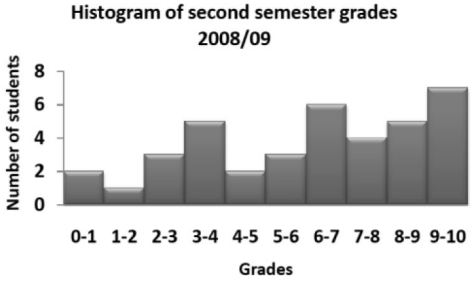


Fig. 4. Dispersion graph of grades obtained in second semester vs. grades obtained in first semester.



(a)



(b)

Fig. 5. Grades histogram for (a) first semester and (b) second semester.

With reference to the distribution of grades, Fig. 5 shows a histogram of grades for the first and second semesters. No grade adjustment has been performed in either case. As can be seen, first semester grades adjust quite well to a normal distribution. This is the natural effect of averaging a large number of partial scores, and inherits the advantages of bell-curve grading (differences in test difficulty are compensated), but not some of the important drawbacks (in this case, grades are not referenced to the performance of the rest of the students in the class).

It is also worth noting that student grades of members who belong to the same team were homogeneous in most cases. In fact, significant differences were only found in two groups. They both decided to communicate the problem after the test for the first learning unit, and ad hoc solutions were adopted in these cases, reallocating team members. To facilitate the integration of rejected students into another team, teams accepting the student were given a bonus. This had validity for a single learning unit, and consisted of considering the grade of the new member only if it was above the average of the grades obtained by all other members in the team.

4.2 Student questionnaire

To derive useful information about the student perception of the methodology, they were ask to fill out an anonymous questionnaire, composed of a series of statements that they had to evaluate on a scale of 1 to 5, with 1 meaning complete disagreement and 5 full agreement with the statement.

The questionnaire was organized into five major sections, as presented below:

Exercise Bulletins:

1. The bulletins help me to prepare the tests for the learning units.
2. I consider that bulletin and exam contents are closely related.
3. Bulletins help me to keep the module up to date.
4. In general, I think that bulletins are a good idea.

Table 1. Average score for each statement in the questionnaire

	Exercise bulletin			Student effort			Team work			Assessment		Teaching method		
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14
Average	4.55	4.65	4.39	3.94	4.13	3.58	3.94	4.03	3.84	4.42	4.52	4.03	4.23	4.29
Std dev.	0.89	0.55	0.72	1.09	0.67	0.96	1.15	0.84	1.07	0.67	0.63	0.75	0.96	0.90

Table 2. Numerical answers in questionnaire

	Average	Std dev.
Number of hours of independent effort	4.00	1.52
Number of times attending tutorials	2.25	2.04
Average time per tutorial (minutes)	42	23

Student Effort:

5. The assessment method used in this module increments the amount of individual effort.
6. In general, I consider that the workload for the module is appropriate.

Team Work:

7. I prefer working in groups rather than individually.
8. The fact that my grade influences other team members' grades has increased my level of commitment to the group.
9. I think that the inclusion of cooperative work in the module has been positive, and that it is a learning method which should be used more intensively during the degree.

Assessment:

10. All questions in the tests were included in the learning objectives for the learning units.
11. By using this assessment system, it is clearer what I have to do to pass the module.

Teaching Method:

12. I consider that lectures and bulletins are well coordinated and that they aid comprehension of the concepts.
13. I think that I learn more and better by using this teaching methodology.
14. In general, I think that the teaching methodology that has been used in this module is adequate.

In addition, students were asked about the number of hours that they dedicated to the module (per week), the number of times that they attended personalized tutorial sessions and the average duration of these sessions.

A total of 31 questionnaires were processed. Table 1 shows the average score and corresponding standard deviations for each statement in the questionnaire. Table 2 shows the same data for questions requiring a numerical answer.

These results show that students value highly both the assessment methodology and the teaching method used in this module; questions relating to these aspects all scored above 4. Questions that relate to learning objectives, exercise bulletins and exam questions (Q1, Q2 and Q10) have also obtained high scores.

Another important result concerns question 6, which received the lowest score overall. Although the average score in this question indicates a degree of disagreement, the number of hours of independent effort indicated by the students exactly matches the equivalent amount of ECTS credits (counting each credit as 30 hours of independent effort). This implies relevant differences between academic guidelines and the student's perception of the amount of effort that they should dedicate to each module.

Each student attended an average of 2.25 personalized tutorial sessions and each session had an average duration of 42 minutes. This implies that each student has benefited from, on average, one and a half hours of individual attention during the semester. This has been possible because of the relatively low number of students on the course.

5. Conclusions

Results from this work suggest that preventing may be more effective than detecting at dealing with the problem of plagiarism. Although detection is an important issue, the use of carefully designed teaching methodologies and assessment strategies may make it unnecessary. In particular, assessment techniques that indirectly reward individual contributions and naturally avoid plagiarism constitute an interesting alternative to others that simply control and punish plagiarism. Furthermore, students' opinions on the methodology reported in this paper indicate an additional positive side effect of this type of strategy, producing an increase in student satisfaction and a decrease of student dropout rates.

In addition, meaningful collaboration can also be easily encouraged by making individual grades dependent on the grades obtained by the rest of the team members. This type of strategy also yields some additional desirable effects such as the achievement of a moral commitment, the establishment of a common goal that benefits cooperative

learning and the encouragement of students to learn from one another.

Acknowledgements—The authors would like to thank the European Convergence office at the University of Valencia (Spain) for the funding provided through Projects DocenTIC and Finestra Oberta 08/DT/04/2009, 18/DT/05/2010 and 47/FO/35/2010.

References

1. C. C. Bonwell and J. Eison, *Active Learning: Creating Excitement in the Classroom. ASHE-ERIC Higher Education Report No. 1*, George Washington University, Washington, DC, Tech. Rep., 1991.
2. M. Prince, Does active learning work? A review of the research, *Journal of Engineering Education*, **93**(3), 2004, pp. 223–231.
3. P. Feden and R. Vogel, *Methods of Teaching: Applying Cognitive Science to Promote Student Learning*, McGraw Hill Higher Education, 2003.
4. S. J. Swan, K. and S. R. Hiltz, Assessment and collaboration in online learning, *Journal of Asynchronous Learning Networks*, **10**(1), 2006, pp. 45–62.
5. D. Johnson, R. Johnson and K. Smith, *Active Learning: Cooperation in the College Classroom*, 2nd edn, Interaction Book Co., Edina, MN, 2006.
6. R. Conway, D. Kember, A. Sivan and M. Wu, Peer assessment of an individual's contribution to a group project, *Assessment and Evaluation in Higher Education*, **18**(1), 1993, pp. 45–54.
7. S. Divaharan, An attempt to enhance the quality of cooperative learning through peer assessment, *Journal of Educational Enquiry*, **3**(2), 2002, pp. 72–83.
8. E. Tatar and M. Oktay, Relative evaluation system as an obstacle to cooperative learning: the views of lecturers in a science education department, *International Journal of Environmental & Science Education*, **3**(2), 2008, pp. 67–73.
9. M. Donnelly, R. Ingalis, T.A. Morse, J. Castner and A.M. Stockdell-Giesler, (Mis)trusting technology that polices integrity: A critical assessment of Turnitin.com, *Inventio*, **1**(8), 2006.
10. J. McCart and J. Jarman, A technological tool to detect plagiarized projects in Microsoft Access, *IEEE Transactions on Education*, **51**(2), 2008, pp. 166–173.
11. F. Rosales, A. Garcia, S. Rodriguez, J. Pedraza, R. Mendez and M. Nieto, Detection of plagiarism in programming assignments, *IEEE Transactions on Education*, **51**(2), 2008, pp. 174–183.
12. B. Scaife, IT consultancy plagiarism detection software report for JISC Advisory Service, NCC Group plc., Manchester, 2007.
13. M. Mozgovoy, T. Kakkonen and G. Cosma, Automatic student plagiarism detection: future perspectives. *Journal of Educational Computing Research*, **43**(4), 2010, pp. 507–527.
14. T. S. Dee and B. A. Jacob, Rational ignorance in education: a field experiment in student plagiarism (Technical Report), National Bureau of Economic Research, ER Working Paper Series, Vol. w15672, January 2010.
15. B. Bolin, Addressing plagiarism with stasis theory, currents in teaching and learning, **2**(2), 2010, pp. 13–21
16. J. Carroll, *A Handbook for Deterring Plagiarism in Higher Education*, 2nd edn, Oxford Centre for Staff and Learning Development, Oxford Brookes University, 2007
17. M. R. Olt, A new design on plagiarism: Developing an instructional design model to deter plagiarism in online courses, Ph.D. thesis, *Dissertation Abstracts International,* **68**(09), 2007
18. M. Hamalainen, Useful tips on avoiding plagiarism, Library Media Connection, **25**(6), 2007, pp. 40–41.
19. B. S. Bloom, *Taxonomy of Educational Objectives*, Allyn and Bacon, Boston, MA, 1984.
20. M. Huyck, D. Ferguson, M. Cama and E. Howard, Work in progress—evaluating the impact of reflective thinking on learning objectives in undergraduate multidisciplinary project teams, *37th Annual Frontiers In Education Conference—Global Engineering: Knowledge Without Borders, Opportunities Without Passports, 2007 (FIE '07)*, October 2007, pp. F4C–26–F4C–27.
21. N. Hoic-Bozic, V. Mornar and I. Boticki, A blended learning approach to course design and implementation, *IEEE Transactions on Education*, **52**(1), 2009, pp. 19–30.
22. N. Linge and D. Parsons, Problem-based learning as an effective tool for teaching computer network design, *IEEE Transactions on Education*, **49**(1), 2006, pp. 5–10.
23. D. Johnson and F. Johnson, *Joining Together. Group Theory and Group Skills*, 10th edn, Pearson, New Jersey, 2009.
24. E. B. Coleman, A. L. Brown and I. D. Rivkin, The effect of instructional explanations on learning from scientific texts. Available at http://www.jstor.org/stable/1466776.
25. J. Holmes, Designing agents to support learning by explaining, *Computers and Education*, **48**(4), 2007, pp. 523–547.
26. R. Dunn and K. Dunn, *Teaching Students Through their Individual Learning Styles: A Practical Approach*. Reston Publishing Company, Reston, VA, 1978.
27. M. Sprenger, *Differentiation Through Learning Styles and Memory*, 2nd edn, Corwin Press, Thousand Oaks, CA, 2008.

Miguel Arevalillo-Herráez received his first degree in computing from the Technical University of Valencia, Spain, in 1993; his BSc in Computing from Liverpool John Moores University, UK, in 1994; and his PgCert in Teaching and Learning in Higher Education and Ph.D. in 1997, both also from Liverpool John Moores University, UK. He was a senior lecturer at this institution until 1999. He then left to work for private industry for a one year period, and came back to the academy in 2000. He was the programme leader for the computing and business degrees at the Mediterranean University of Science and Technology until 2006. Since then, he has lectured at the University of Valencia (Spain), delivering programming and networking modules. His research now concentrates on education and applied artificial intelligence.

José M. Claver received his M.Sc. in physics from the University of Valencia, Burjassot, Spain, in 1984, and his Ph.D. in computer science from the Technical University of Valencia, Spain, in 1998. From 1985 to 1990 he was with the Electronics and Computer Architecture Department, University of Castilla-La Mancha, Albacete, Spain. From 1991 to 2008, he was with the Department of Computer Science, Jaume I University, Castellón, Spain. Since 2007, he has been an Associate Professor at the Computing department of the University of Valencia. He has taught undergraduate courses on computer architecture and embedded systems and graduate courses on high-speed networks, advanced computer architecture, and parallel computing. His research interests include education, computer architecture, parallel computing, high-speed QoS networks, network protocols, embedded systems, and reconfigurable computing. He is the author or co-author of more than forty research publications on these subjects.