SIMPROGRAMMING: THE DEVELOPMENT OF AN INTEGRATED TEACHING APPROACH FOR COMPUTER PROGRAMMING IN HIGHER EDUCATION

Daniela Pedrosa[1,2], José Cravino[1,2], Leonel Morgado[3,4], Carlos Barreira[5], Ricardo Rodrigues Nunes[1,4], Paulo Martins[1,4], Hugo Paredes[1,4]

[1] Universidade de Trás-os-Montes e Alto Douro (UTAD), Vila Real (PORTUGAL)
[2] Research Centre "Didactics and Technology in Education of Trainers", Aveiro (PORTUGAL)
[3]Universidade Aberta, Coimbra (PORTUGAL)
[4] INESC TEC, Porto (PORTUGAL)
[5] Faculdade de Psicologia e Ciências da Educação da Universidade de Coimbra (PORTUGAL)

Abstract

Computer programming courses in higher education tend to have high rates of academic failure and students struggle, particularly so in the transition from entry-level programming to advanced programming. Some of the reasons given in the literature relate to the type of teaching approach and the strategies used by students and their attitudes towards computer programming. The literature also mentions that educational approaches are not always appropriate to the needs of students and to the development of skills required in the job market.

We developed a teaching approach to try to address some of these issues and support students learning computer programming in the transition from entry-level to advanced computer programming: the SimProgramming approach. This approach was introduced at the University of Trás-os-Montes e Alto Douro (Portugal), within the scope of the course "Programming Methodologies III", part of the second curricular year of the programmes of studies in Informatics Engineering and in Information & Communication Technologies.

We present in detail the origins of the SimProgramming approach, starting from the first trials that introduced, in two iterations, learning activities based on problem-based learning, and up to the third iteration where the current SimProgramming approach was implemented. We describe the reasoning, design and implementation of these three iterations, to show how the approach evolved.

The SimProgramming approach is based in four conceptual foundations: business-like learning environment, self-regulated learning, co-regulated learning and formative assessment. For each of these conceptual foundations, we explain the teaching strategies adopted. In SimProgramming, the learning activity process develops in four phases, and students have specific tasks in each phase.

We analyse interview data regarding student perceptions about the SimProgramming approach, and registration grids data on team work dynamics and final assessment of the assignment, noting the impact of SimProgramming in student grades.

The application of SimProgramming revealed promising evidences in the overall results of student learning in the activities proposed in this approach. The average grades improved, and did the number of students regularly submitting their tasks on schedule. The perceptions of students regarding the SimProgramming approach are very positive: they recommend using it in the following years, and provided some suggestions to improve the approach.

We conclude with reflections and recommendations for subsequent development of the SimProgramming approach in its application to the teaching of computer programming and potential for using it in other educational contexts.

Keywords: Computer Programming; Development of teaching approach; Teaching Strategies.

1 INTRODUCTION

Teaching of computer programming has been the subject of several research efforts, due to the complexity of the courses [1], students' difficulty in learning to program [2], and lack of motivation and involvement of students in study [1][2][3][4][5][6], leading many to abandon academia or pursue

professional paths that do not involve programming [7]. Even among students completing programs with success, most come to the job market unprepared and lacking the necessary skills to meet the expectations of employers [4], including teamwork and cooperation skills [3].

There are a number of reasons for this problem, such as [8]: inadequate teaching methods; students' frequent use of study methods that are not suitable for learning programming; students' abilities and attitudes; psychological effects, and the nature of programming itself. Also, current learning approaches are not in line with the professional practice required of students by the job market [9].

In advanced programming courses the level of complexity is much greater than entry-level programming courses. For example, the students have difficulties learning Object-Oriented Programming, not only because it requires the understanding of abstract concepts [10] but also because students have not encountered more complex programming situations where object-orientation can provide tangible benefits. Situations where large code sizes, team dimension hinders communication or regular changes to existing code are necessary. To use architectural styles such as Model–View–Controller (MVC) [11], students need to develop complex skills in programming [12] and develop social skills [3]. In fact, the pedagogical context in which students learn influences their engagement and resolve to achieve learning outcomes [13]. There is a need to adapt curricula to new pedagogical developments [14].

In this work we aim to support the teaching-learning process of computer programming in the transition of students from entry-level programming to advanced programming. Using Design Science research [15], we developed a learning approach based on problem-based learning – (PBL) [16], which was reshaped and improved throughout three iterations [5][6]. In each iteration, new activity plans were designed, alongside with resources and tools. Data was collected, and upon its analysis and reflection, changes to learning strategies and interventions were implemented. In the 3rd iteration we labelled this approach the SimProgramming approach, and we describe it in this paper.

2 BACKGROUND

The challenge of teaching programming is complex, because it depends on a diversity of factors, including: programming experience and skills of the teacher, pedagogical approach, and type of tools used [4]. It is difficult for the teacher to provide a learning environment that benefits all students, due to their different attitudes towards the learning of programming. Thus, it is advisable to use other complementary learning activities [17] to stimulate the interest and active involvement of students throughout a course as appropriate to the goals, resources, assessment, and feedback [1].

Students in active learning discover and construct knowledge through active participation and engagement in the learning process, through meaningful activities [18]. Active participation is particularly ensured when students have the ability to successfully apply self-regulation strategies, including goal setting, selection and implementation of strategies, and self-efficacy [19], leading to more time and energy investment [20]. Self-regulation involves an interplay between commitment, control, and confidence, actively monitoring the various processes of learning [21]. When students have the ability to successfully use self-regulation strategies, they are motivated to actively participate in conclusion out their academic tasks [22] [19].

Evaluation and effective formative feedback help self-regulation by supporting motivation [19] and enabling self-awareness of work developments [23]. Self-regulation is also a major variable influencing academic procrastination and performance [24],a recurrent phenomenon connected to time management, contributing to delay of completion or initiation of academic activities (ibid.). Students experience it throughout their academic career [25], despite expectations of worse outcomes on account of it [26]. It is also associated with higher levels of stress and anxiety [27].

Self-regulation is one of the keys to understanding procrastination as well as self-efficacy [28]. Also, procrastination is associated with imbalances in self-regulation due to concerns about incompetence and failure: the delay of the delivery of work, may reflect performance-related apprehension on the part of students, seen as a minor encumbrance, sufficient to attain a "good" performance relative to their peers [29]. Students who have self-efficacy for self-regulation of learning know how to employ their self-knowledge to manage their learning and their commitment to meet challenges [30].

The self-evaluation provides information for students to analyze how they learned and understood the objectives of the learning process. So students being able to make an effective self-evaluation is necessary to create conditions for success [30], including: raising awareness on the value of self-

assessment; access to the evaluation criteria of a specific task or performance to be evaluated; direct instruction; feedback; self-assessment practices; and opportunities to improve the task performance.

Currently, we witness a considerable increase in the number of research efforts which follow the idea that social context and social aspects are relevant for self-regulation of student learning [31][32]. The above-mentioned personal learning perspectives are complemented with the social learning context of peers and teachers [23], which form a community of practice – more functional and effective if sharing a passion for what they do and interact regularly to improve it [33].

The "Socially shared regulation of learning" (SSRL) refers to when a group builds and shares perceptions on a set of tasks and goals to be achieved [32]. Also, it involves shared regulatory processes based on knowledge beliefs (such as motivation strategies, decision-reviewed monitoring and goals), which compose a result of developed co-learning [32].

In collaborative work, the way students are grouped has impacts in the results of the learning process. Coordination is critical for the success of collaborative learning: teamwork is effective when there is allocation of responsibilities and roles to specific team members [3]. PBL promotes teamwork to discover or propose a solution to a specific problem. A team leader is essential to facilitate the integration of information and to guide the team through the learning process [3].

3 THE SIMPROGRAMMING APPROACH: DEVELOPMENT PROCESS

We applied this process throughout three years (2010-2013) at the University of Trás-os-Montes e Alto Douro (UTAD), Portugal, within the scope of the course "Programming Methodologies III" (PM3), a mandatory unit in the second curricular year of the undergraduate study programmes in Informatics Engineering (IE) and in Information & Communication Technologies (ICT), which introduces concepts of software architecture to support students' development in their code organization skills.

To understand and improve the learning process of students, we designed a learning activity through Design Science [15], a dynamic process of planning/designing, prototyping, test and analysis, and reflection over the implemented strategies during two iterations (PM3 in 2010/2011; and PM3 in 2011/2012), which led to the 3rd iteration, emerged the SimProgramming[1] approach (figure 1).

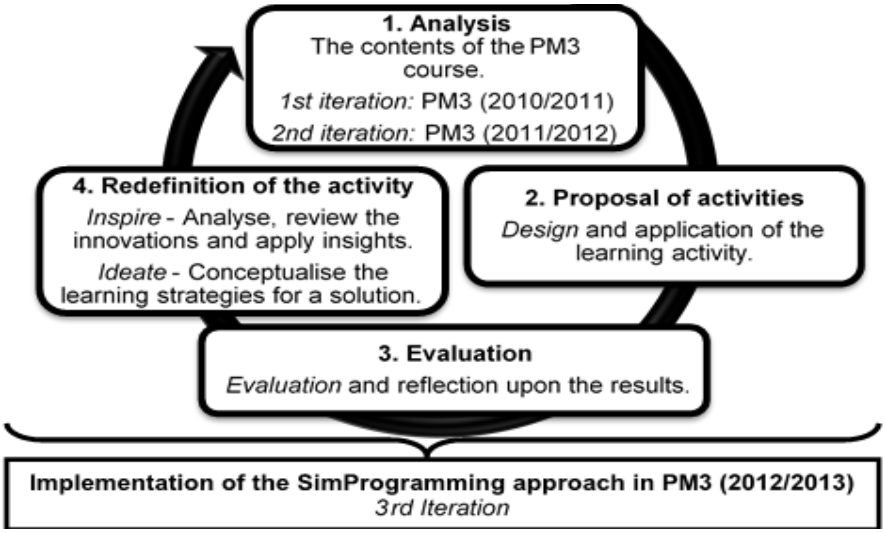


Fig. 1: The origin of the SimProgramming approach

This dynamic process developed in four cyclical phases: the first phase consisted on *Analysis* of the contents of the PM3 course; the second phase was the *Proposal of activities* to students, by design and application (prototyping) of the learning activity; the third phase was the *Evaluation* and reflection on the activity process and results; finally, the fourth phase was devoted to the *Redefinition of the activity*, by analysing and reviewing the innovations, and applying insights from that analysis/review

[1] The designation of the "SimProgramming" was created by the teacher and the team tutors the PM3. The "SimProgramming" results of a play on words that has two understandings: 1st. "Sim" in Portuguese means "yes", in this sense, is a positive reinforcement to programming; and 2nd: "Sim" derives from the word "simulation" and taking as inspiration the game "Sims".

and from literature, to create the learning strategies to be used as the updated prototype for the following iteration. In the third iteration, we gave the current prototype its current name, hence this was the *Implementation of the SimProgramming* approach in PM3 (2012/2013).

The learning activity is based on problem-based learning. We assigned to each team a specific problem involving a MVC-related software architecture in order to stimulate and foster advanced programming skills in students. Students had to develop a written document, explaining in detail of the coding approaches they used to apply their assigned architecture to the problem involving different frameworks, libraries, and/or specific APIs.

3.1 What were the *reasons* for the origin the SimProgramming approach?

The first iteration, detailed on an earlier paper [5], took place during the 2nd semester of the 2010/2011 academic year. It resorted to programming assignments with a theory component, and required the involvement of students in online communities of practice, where professionals and practitioners were present. The goal was for students to develop problem-analysis competences but enable them to realize the relevance of these competences from the perspective of professionals and practitioners as a source of motivation. The results of this first iteration have shown that most groups of students were unable to solve their assignments successfully (ibid.), and that the overall quality of the work was poor.

The second iteration took place the following year (2011/2012). It was also detailed in an earlier paper [6]. The activity was deployed with new pedagogic strategies: the components of the activity were more structured in time (weekly tasks and deadlines), there were two tutors to provide students with support, monitoring, and feedback, and three group dynamics were conducted within an auditorium. More groups completed the activity in this second iteration, but the overall quality of student work remained poor.

In both iterations, the lack of motivation, the lack of feedback on the development of the work and lack of time were identified as the main problems. Thus, in the third iteration (2012/2013), the activity was again reshaped, taking on its current “SimProgramming” form.

3.2 The SimProgramming: Conceptual foundations

3.2.1 Conceptual foundation 1: Business-like

This focuses on students’ the lack of touch with professional reality and expectations, including teamwork and cooperation. It employs PBL, with teamwork to promote the collaborative discovery or proposal of a solution to a problem, and a group leader to facilitate the integration of information and guide the group [3]. In this sense, the SimProgramming simulates a business-like environment, with each participant taking on a role.

The course Professor plays the role of general manager, globally in charge, including course content and monitoring. Course tutors or teaching assistants take on the role of project managers, doing close monitoring, mentoring, and providing feedback, based on the Scrum method for project management and agile software development [33].

Project Managers are available for unscheduled face-to-face support, and conduct scheduled weekly meetings with team leaders, reviewing three topics: 1) what did you do last week?; 2) what will you do this week?; and 3) what is preventing task completion? Upon detecting specific issues (technical, personal, or others), they set up focused 30-minute meetings with the involved team.

Students form development teams (15 teams of 7 students). The team divides the work according to the role played by each member: one student as team leader and remaining students handling subsets of work (packages). Each student needs to master his/her package and the team leader needs to make sure members keep a global view of the project context and status, integrating knowledge.

3.2.2 Conceptual foundations 2: Self-regulated learning

Self-regulation as a motivational key to active participation and engagement of students on meaningful activities. It implies active participation of students before, during, and after completion of academic work [19]. SimProgramming promotes active learning and student self-regulation of learning, by focusing team members on research and exploration tasks of the assigned problem/packages, not just

on development. That is, students have to both solve their individual package and contribute to the overall perspective of the team problem. Explicitly, the leader needs to integrate the research and exploration output of members both to report at weekly project management meetings and ensure information flow within the team. Weekly forms allow students to self-reflect upon their work, ponder on what to do the following week, and reflect upon the factors that prevent them from achieving the team and individual objectives. Students thus have to develop self-regulation skills to balance the SimProgramming objectives with their individual goals.

Following the background in time management and procrastination, the SimProgramming approach includes encouragement of students to adopt study routines, by creating a context where tasks are performed continuously: students are strongly encouraged to contribute to their project on a weekly basis: this connects with the above conceptual foundation on self-regulated learning, in that feedback and monitoring need to be approached in support of self-reflection and self-regulation, not as mere checkpoints or status controls. Throughout the learning process the aim is for students to gradually develop the concept of having to do their work regularly and not only at the last moment.

3.2.3 Conceptual foundation 3: Co-regulated learning

Throughout the activity, students were performing team tasks, such as reports and presentations about the work. In the SimProgramming, this aims to support the functional and effective development of the learning community of practice around problem-solving. For this, there is both encouragement and support for social involvement of students in pre-existing online communities of professionals on the field, and with former students, to discuss the technologies under study or the profession [5].

The contact with tutors, in meetings, classes, and other methods (e.g., on-line), also aims to stimulate students' demand for social help (peers, teachers, tutors, etc.) to clarify their doubts and difficulties. Management (tutors and professor) provide this support by advising on methods of gradual participation and involvement in communities, including suggestion of specific tasks for clarification. As a vehicle for more frequent and more homogenous peer-based contributions and discussion, supporting community development, informal interactions and debate were promoted and monitored via a Facebook group for the course. It was also a forum which management used to have teams discuss their problems with each other. This has been maintained over the years, and is now a source of interaction between new students, former students of the course (still undergraduates or graduate students), and former students that went on to become professional developers. Some of these former students offered to play the role of business consultants, providing support and advice to teams.

3.2.4 Conceptual foundation 4: Formative Assessment

Formative Assessment is provided by management (tutors and professor), both face-to-face and online, based on monitoring, meetings, and social media interactions. This includes motivational mentoring, and feedback on individual package status (e.g., work progressing or deviating from expectations and goals). But critically, it also leverages the weekly forms mentioned above, by employing them as an opportunity to provide feedback on students' own assessment and cognitive perspective of the task – feedback in support of self-regulation and critical thinking [22]. Also, in SimProgramming we stipulate self-assessment of individual students and hetero-assessment by team members in the final assignment, for comparative analysis between students' grading expectations and final grades.

3.3 SimProgramming phases: learning activity process

The SimProgramming approach develops the learning activity process along four phases, based on the above conceptual foundations.

In first phase, over 3 weeks, each team searches for information on the topic of their problem in a variety of sources: textbooks, technical manuals and documentation, blogs, forums, and scientific-technical papers. They also establish contact with online communities of practice and start to interact there informally (not yet addressing their assigned problem). Teams are established and problems assigned, tasks/packages are distributed among team members, and finally the team leader is elected. The purpose of this phase is for the teams to become acquainted with the technologies associated to their problem and to start developing online social interaction competences. At the end of this phase, groups present the status of their efforts, i.e., information retrieved on the individual

technologies impacting their problem (including code samples) and their preliminary analyses of that material.

In second phase, over 4 weeks, the students start to attempt integration of technologies towards solving their problem. There are on-going interactions in online communities and information searching. At the end of this stage, students again present the status of their efforts. They are expected to have adequate theoretical knowledge about the involved technologies and some level of coding skills with them, being able to propose a solution for the problem with concrete code examples, prototypes or proofs-of-concept.

The third phase (2 weeks) ensues, aiming to improve students' results from the second phase, following feedback by management (professor and tutors). At the end of this phase, teams present the result of their efforts throughout the semester. This presentation includes a reflection upon the problems they encountered, an explanation of their approach for technology integration and the proposed solution to their assigned problem.

The fourth and final phase is a catch-up phase for teams who for some reason were unable to perform satisfactorily in prior phases, or simply wish to improve their results, in 2 weeks. This phase is particularly important in the learning design to account for students who may face exceptional personal circumstances during the semester.

Throughout all phases, weekly meetings take place between project managers (tutors) and team leaders, providing feedback for motivation, self-regulation and possible support for technical doubts. When internal team disruptions or high-risk situations are detected, project managers may also schedule meetings with those specific teams or students, for targeted intervention.

4 DATA COLLECTION

In order to verify whether any improvements occurred, during the use of the SimProgramming approach, in overall results in the teams and the students, compared with the previous trials, we look at the overall data on the average grades of activity and team work dynamics (regular delivery of tasks) obtained by the evaluation and observation grids.

We conducted participant observation [34] and the evaluation and observation grids consist of the field notes [34]. These grids are organized in tables as such: notes about the weekly forms, to record if the tasks were delivered, if the student has reached the week's goals, whether tasks are on track and which aspects need improvement; notes about task groups with same goal; about team dynamics (role changes, student withdrawal); and notes about which teams received tutor intervention.

Also, we conducted 21 semi-structured interviews [34], and prepared an interview guide with 5 groups of questions. One of question groups was about the perceptions of students about the SimProgramming approach. The aim was to collect student opinions about SimProgramming in their learning and suggestions for improvement in future iterations.

We selected students based on different learning behaviors: role in the team (e.g. team leader), results in the assignment, special status (e.g. working-students), and students which the quality increased the working strategies during the assignment.

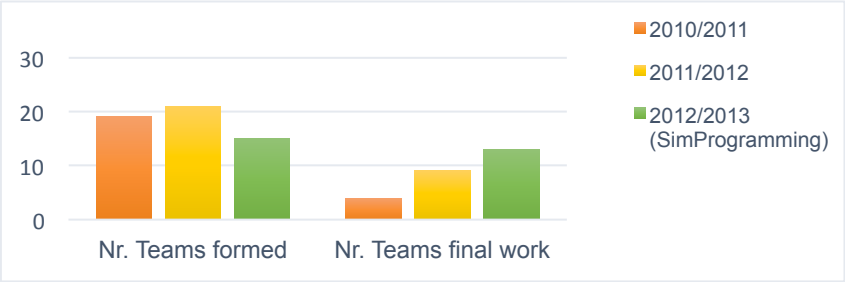
The procedure adopted for thematic analysis [35], was the construction of content analysis matrixes organized into categories, subcategories, indicators, and recording units, which have been restated during the process of content analysis. Then we conducted a cyclical process of improvement, synthesis, and reflection on the results.

5 RESULTS AND DISCUSSION

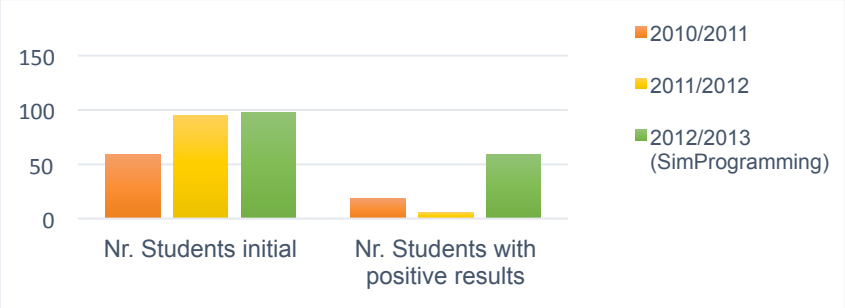
5.1 Overall SimProgramming assignment results

Before introducing SimProgramming, in PM3 2010/2011, the first research iteration, of from the 19 teams only 7 showed some results during the design phase and 4 teams obtained positive results. Where most of the students did not get positive results, in 59 students only 18 obtained positive results [5]. In PM3 2011/2012, the second research iteration, only 9 of 21 teams work actually started and completed the project, in 95 students only 6 students positive result [6].

With SimProgramming, in PM3 2012/2013, the third research iteration, we observed compared to previous trials ([5][6]).More teams concluded the assignment and work grades improved, in 15 teams, 11 successfully achieved the learning goals of SimProgramming (Graphic 1). Two other teams completed the requested tasks but the quality of their work was poor. The remaining two teams never actually started. And in 97 students 59 obtained positive results (Graphic 2).



Graphic 1



Graphic 2

The 68% of students who completed the activity, much more than in previous trials (Table 1). In 13 teams, 8 teams attained above 15 values out of 20, which we believe is a good result. Most teams maintained a regular pace of work delivery for both individual and team work (10 teams). But there were exceptions: some teams (teams B, E, I) in a week did not deliver the requested tasks.

In some teams there was change in its dynamics, namely: team leader change and students dropping out of the activity (teams D, E, N). The teams who had the intervention of the tutors were able to complete the activity with satisfactory results, except team C.

Table 1: Overall results of the evaluation teams

Team	**Nr. Students with a final classification**	**Average grades** (0-20)	**Comments**
A	*6/6*	*12,4*	- Were regular in delivery of individual tasks and teamwork.
B	*7/7*	*16,2*	- Were scheduled to deliver the forms except in the final 2 weeks. They provided all group work.
C	*6/7*	*8,2*	- They were not regular in individual work. Did not hand up the group work, just two reports. - A student gave up on the activity, and two other students attained negative grades. - Intervention of the tutors through meetings providing feedback and motivation.

D	*4/7*	*13,8*	- They were not regular on individual tasks. Performed all group activities, except for the first presentation. - The team changed (some students gave up on the activity). - Intervention of the tutors through meetings providing feedback and motivation.
E	*6/7*	*15,4*	- Were regular delivering individual tasks and teamwork, but didn't undergo the first two presentations. - Initially the team was not achieving the activity goals, but after changing the team leader the team showed good results. - The student who dropped out of the activity was the former team leader. - Intervention of the tutors through meetings providing feedback and motivation.
F	*4/4*	*15,5*	- Were regular delivering individual tasks and teamwork. Team members were working students. - Intervention of the tutors through meetings providing feedback and motivation.
G	*0/6*	-	- The team gave up during the second week of activity. - Students did not respond to meeting tutors' requests.
H	*7/7*	*15,7*	- Were regular delivering individual tasks and teamwork.
I	*5/6*	*15,1*	- Were regular delivering individual tasks and teamwork, except on the last week of the activity. - The team had a dedicated and committed team leader, which was relevant to the team's success.
J	*4/6*	*15,3*	- Were regular delivering individual tasks and teamwork.
K	*0/6*	-	- The team did not work. Students did not respond to the meeting tutors' requests.
L	*2/7*	*4,1*	- The team did not work. - Most team members did not respond to the meeting tutors' requests. Only two students responded and were given an extra activity to compensate.
M	*7/7*	*10,2*	- Were regular delivering individual tasks and teamwork.
N	*3/8*	*15,5*	- Were regular delivering in individual tasks and teamwork. - The team changed (some students gave up on the activity). - Intervention of the tutors through meetings providing feedback and motivation.
O	*4/5*	*15,9*	- Were regular delivering individual tasks and teamwork.
Total	*66/97 (68%)*	*13,3*	- 66 of 97 students who registered for the activity obtained final classification (68% of students), and 59 students obtained positive results.

5.2 Perceptions student´s about SimProgramming approach

Nine students interviewed consider that despite filling weekly individual forms was work intensive and repetitive, but 13 students mentioned they were useful in planning the tasks, organization and compilation of material, promoting regular work and delivery of tasks, and providing feedback to the team (2 students) and to tutors (7 students). Sample quotes from students:

> *"Useful, makes us deliver something at that time (...) Only makes us to study." (E4, 20/05/2013)*

> *"I think it is so good for giving us a sense of the work we did, of what we can do next week" (E18a, 29/05/2013)*

> *"It has its importance overall and also to those who follow us to see if we have done something. (...) Then you know what each one is doing" (E6, 21/05/2013)*

Students gave some improvement suggestions, one them is that weekly individual forms occur fortnightly instead of weekly and realized in team or replaced by meetings. One student mentioned that SimProgramming approach contributed to the preparation of study for the theoretical written tests of the course, and two students mentioned the development of new knowledge and improving grades:

> *"for the theoretical part, we have it that the SimProgramming (...) still makes us to read a lot about the things that will be in the test " (E17a, 23/05/2013)*
>
> *"It is a good opportunity to develop this new knowledge and helps the final grade" (E21a, 05/06/2013)*
>
> *"(…) the grades of PM3 made me feel like it was getting better." (E6, 21/05/2013).*

6 CONCLUSIONS

In the SimProgramming approach, the results in assignment are very positive: more teams completed the assignment and more students obtained positive results in assignment compared, than in previous trials. Furthermore, most of the teams (8 teams) achieved grades equal or above 15 (out of 20). In this context these are quite satisfactory. In the students' opinion the SimProgramming approach was a positive experience, but may benefit from some adjustments in certain tasks.

SimProgramming approach simulated a business environment, with teamwork, leaders and roles. Aspects such as the relational dynamic of the team, the type of team leader, working strategies, proved to be essential to the progress of students work. Coaching and feedback fundamentally allowed to set teams on new paths, encouraging participation and student involvement towards completion of the assignment. Weekly meetings with team leaders were key moments enabling quick follow-up in a way that wouldn't be viable individually with all 97 enrolled students. The weekly individual forms provide a variety of information that is crucial for the teaching team.

Tutors were involved in assisting the lecturer and the teaching assistant in charge of lessons, to perform close monitoring of students' progress that is fundamental to this approach, implying that it requires significant changes to the traditional college-level teacher-student relationships, class schedules, and profile of teaching activities. The feedback provided by tutors, in the students' perspective, helped their motivation and contributed to improve their work.

We believe that SimProgramming may also hold potential for other educational contexts where the transition from early to advanced skills is key. Hence we suggest further research into the development of strategies of self-regulation and co-regulation of learning.

ACKNOWLEDGMENTS

Pedrosa, D. and Nunes, R. R. thank the Fundação para a Ciência e Tecnologia (FCT), Portugal, for Ph.D. Grants SFRH/BD/87815/2012 and SFRH/BD/91309/2012. To all the students and teachers who collaborated on the research.

REFERENCES

[1] Robins, A., Rountree, J. & Rountree. N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, 13(2), 137–172.

[2] Lahtinen, E., Ala-Mutka, K., & Järvinen, H. M. (2005). A study of the difficulties of novice programmers. In *ACM SIGCSE Bulletin*, 37(3), 14-18.

[3] Sancho, P., Moreno-Ger, P., Fuentes-Fernández, R., & Fernández-Manjón, B. (2009). Adaptive Role Playing Games: Na Immersive Approach for Problem Based Learning. *Educational Technology & Society*, 12(4), 110–124.

[4] Kumar, B. (2012). Gamification in Education-Learn Computer Programming with Fun. *International journal of computers & distributed systems*, 2(1), 46-53.

[5] Morgado, L.; Fonseca, B.; Martins, P.; Paredes, H.; Cruz, G.; Maia, A. M.; Nunes, R.; Santos, A. (2012). Social networks, microblogging, virtual worlds, and Web 2.0 in the teaching of programing techniques for software engineering: a trial combining collaboration and social interaction beyond college. In *Global Engineering Education Conference (EDUCON)*.1-7. IEEE. doi:10.1109/EDUCON.2012.6201129.

[6] Nunes, R. R., Pedrosa, D., Fonseca, B., Paredes, H., Cravino, J., Morgado, L., & Martins, P. (2015). Enhancing students' motivation to learn software engineering programming techniques: a collaborative and social interaction approach. In *Universal Access in Human-Computer Interaction. Access to Learning, Health and Well-Being*. 189-201. *Springer International Publishing*.

[7] Lethbridge, T. C., Diaz-Herrera, J., LeBlanc Jr, R. J., & Thompson, J. B. (2007). Improving software practice through education: Challenges and future trends. In *Future of Software Engineering, 2007. FOSE'07*, 12-28. IEEE.

[8] Gomes, A., & Mendes, A. J. (2007). Learning to program-difficulties and solutions. In *International Conference on Engineering Education–ICEE*, 2007, Coimbra, Portugal.

[9] Sheppard, S. D., Macatangay, K., Colby, A., & Sullivan, W. M. (2008). *Educating Engineers: Designing for the Future of the Field*. Book Highlights. Standford: Carnegie Foundation for the Advancement of Teaching.

[10] Allinjawi, A. A., Al-Nuaim, H. A., & Krause, P. (2014). Evaluating the Effectiveness of a 3D Visualization Environment While Learning Object Oriented Programming. Journal of Information Technology and Application in Education, Vol. 3. doi: 10.14355/jitae.2014.0302.01

[11] Curry, E., & Grace, P. (2008). Flexible self-management using the model-view-controller pattern. *Software*, IEEE, 25(3), 84-90.

[12] Jenkins, T. (2002). On the difficulty of learning to program. In *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, 4, 53-58.

[13] Johri, A., & Olds, B. M. (2011). Situated engineering learning: Bridging engineering education research and the learning sciences. *Journal of Engineering Education*, 100(1), 151-185.

[14] Association for Computing Machinery (ACM) & IEEE Computer Society (IEEE-CS), 2008. "Computer Science Curriculum 2008: An Interim Revision of CS 2001." *Computing Curriculum Series* [on-line], http://www.acm.org//education/curricula/ComputerScience2008.pdf.

[15] Hevner, A., & Chatterjee, S. (2010). Design science research in information systems. *In Design Research in information systems,* (pp. 9-22). Springer science + Business Media. USA.

[16] Savery, J. R. (2015). Overview of problem-based learning: Definitions and distinctions. In Walker, A.; Leary, H.; Hmelo. Silver, C.; Ertmer, P. (Eds), *Essential Readings in Problem-Based Learning, (pp. 5-16). Purdue University Press. Indiana, USA.*

[17] Chen, G. D., Li, L. Y., & Wang, C. Y. (2012). A Community of Practice Approach to Learning Programming. *Turkish Online Journal of Educational Technology-TOJET*, *11*(2), 15-26.

[18] Alharbi, A., Paul, D., Henskens, F. & Hannaford, M. (2011). An investigation into the learning styles and self-regulated learning strategies for Computer Science students. In G.Williams, P. Statham, N. Brown & B. Cleland (Eds.) *Changing Demands, Changing Directions*. Proceedings ascilite Hobart 2011. (pp. 36-46).

[19] Clark, I. (2012). Formative Assessment: Assessment Is for Self-regulated Learning. Educational Psychology Review, 24(2), 205–249.

[20] Zimmerman, B. J. (2008). Investigating Self-Regulation and Motivation: Historical Background, Methodological Developments, and Future Prospects. *American Educational Research Journal*, 45(1), 166–183.

[21] Nicol, D. J., & Macfarlane-Dick, D. (2006). Formative assessment and self-regulated learning: a model and seven principles of good feedback practice. *Studies in Higher Education*, 31(2), 199–218.

[22] Greene, J. A., & Azevedo, R. (2007). A theoretical review of Winne and Hadwin's model of self-regulated learning: New perspectives and directions. *Review of Educational Research*, 77(3), 334-372.

[23] Ozogul, G., & Sullivan, H. (2007). Student performance and attitudes under formative evaluation by teacher, self and peer evaluators. *Educational Technology Research and Development*, 57(3), 393–410.

[24] Michinov, N., Brunot, S., Le Bohec, O., Juhel, J., & Delaval, M. (2011). Procrastination, participation, and performance in online learning environments. *Computers & Education*, 56(1), 243–252.

[25] Ferrari, J. R., O'Callaghan, J., & Newbegin, I. (2005). Prevalence of procrastination in the United States, United Kingdom, and Australia: Arousal and avoidance delays among adults. *North American Journal of Psychology*, *7*(1), 1-6.

[26] Tan, C. X., Ang, R. P., Klassen, R. M., Yeo, L. S., Wong, I. Y. F., Huan, V. S., & Chong, W. H. (2008). Correlates of Academic Procrastination and Students' Grade Goals. *Current Psychology*, 27(2), 135–144.

[27] Howell, A. J., & Watson, D. C. (2007). Procrastination: Associations with achievement goal orientation and learning strategies. *Personality and Individual Differences*, 43(1), 167-178.

[28] Klassen, R. M., Krawchuk, L. L., & Rajani, S. (2008). Academic procrastination of undergraduates: Low self-efficacy to self-regulate predicts higher levels of procrastination. *Contemporary Educational Psychology*, *33*(4), 915-931.

[29] Andrade, H. (2010). Students as the definitive source of formative assessment: academic self-assessment and self-regulation of learning. In H. Andrade e G. Cizek (eds). *Handbook of Formative Assessment*, (pp.90-105). New York: Routledge.

[30] Hadwin, A. F., Järvelä, S., & Miller, M. (2011). Self-regulated, co-regulated, and socially shared regulation of learning. In B. J. Zimmerman & D. H. Schunk (Eds.), *Handbook of selfregulation of learning and performance,* (pp.65-84). New York: Routledge.

[31] Panadero, E., & Järvelä, S. (2015). Socially shared regulation of learning: A review. *European Psychologist.* doi: 10.1027/1016-9040/a000226.

[32] Wenger, E. (2000). Communities of practice and social learning systems. *Organization*, 7(2), 225-246.

[33] Schwaber, Ken (2004). Agile project management with Scrum. Microsoft Press.

[34] Cohen, L; Manion, L; Morrison, K. (2011). *Research Methods in Education*. 7th Edition. London, Routledge-Taylor & Francis Group.

[35] Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative research in psychology*, *3*(2), 77-101.