Self-regulated Learning in Computer Programming: Strategies Students Adopted During an Assignment

Daniela Pedrosa[1,2(✉)], José Cravino[1,2], Leonel Morgado[3,4], and Carlos Barreira[5]

[1] Universidade de Trás-os-Montes e Alto Douro (UTAD), Vila Real, Portugal
{dpedrosa, jcravino}@utad.pt
[2] Research Centre "Didactics and Technology in Education of Trainers", Aveiro, Portugal
[3] Universidade Aberta, Coimbra, Portugal
leonel.morgado@uab.pt
[4] INESC TEC, Porto, Portugal
[5] Faculdade de Psicologia e Ciências da Educação da Universidade de Coimbra, Coimbra, Portugal
cabarreira@fpce.uc.pt

Abstract. The SimProgramming teaching approach has the goal to help students overcome their learning difficulties in the transition from entry-level to advanced computer programming and prepare them for real-world labour environments, adopting learning strategies. It immerses learners in a business-like learning environment, where students develop a problem-based learning activity with a specific set of tasks, one of which is filling weekly individual forms.

We conducted thematic analysis of 401 weekly forms, to identify the students' strategies for self-regulation of learning during assignment. The students are adopting different strategies in each phase of the approach. The early phases are devoted to organization and planning, later phases focus on applying theoretical knowledge and hands-on programming. Based on the results, we recommend the development of educational practices to help students conduct self-reflection of their performance during tasks.

Keywords: Self-regulation learning · Computer programming · Self-regulated learning strategies

1 Introduction

In higher education, high rates of academic failure and students' difficulty in learning to program are common in computer programming courses [1], particularly in the transition from entry-level programming to advanced programming. Reasons include the teaching approach and the attitudes/strategies used by students in computer programming [2], and lack of motivation and involvement in study [3, 4]. After graduation,

© Springer International Publishing Switzerland 2016
C. Allison et al. (Eds.): iLRN 2016, CCIS 621, pp. 87–101, 2016.
DOI: 10.1007/978-3-319-41769-1_7

most students come to the job market lacking necessary skills to meet the expectations of employers [5], such as: teamwork and cooperation skills [6].

In advanced programming courses the level of complexity is much greater than entry-level programming courses. For instance, students have difficulties learning in situations involving large code sizes, where the team dimension hinders communication or regular changes to existing code become necessary. When applying architectural styles such as Model–View–Controller (MVC) [7], students have difficulties grasping the rationale of architectural styles and other software engineering concepts [8]. Further, students need to develop complex programming skills [9] and social skills [6].

In higher education, one key element is self-regulated learning (SRL), which allows students to be proactive and manage their learning and development of life skills [10]. The application of SRL strategies typically predicts high academic achievement [11]. SRL processes can be improved with appropriate interventions [12], and it is typically recommended that teachers contribute and promote students' development of metacognitive knowledge about academic work and task-specific strategies [13].

In computer science, students that apply SRL and metacognitive strategies exhibit good performance [14]. Often students are not aware of SRL and metacognitive strategies that can be used, so instilling them is important [15].

We developed the SimProgramming approach [16], and applied it in the academic year 2012/2013 to the Programming Methods 3 (PM3) course, part of the second year of the bachelor programmes in Informatics Engineering (IE) and Information & Communication Technologies (ICT) at the University of Trás-os-Montes e Alto Douro (UTAD), Portugal. In the SimProgramming approach, students develop a problem-based learning (PBL) activity within the syllabus of the course, with a specific set of tasks based on the conceptual foundations of SimProgramming. One of these tasks is the filling of weekly forms, which are handed in by each student with a self-reflection on performance of the weekly development of the course assignment. We conducted thematic analysis of the 401 weekly forms to identify the SRL strategies mentioned by students during the development of the assignment.

2 Background

SRL is seen as the students' proactive and intentional monitoring of their actions, adapting and regulating cognition, behavior, emotions, and motivation using personal strategies to enhance learning processes and achieve personal goals [17–20].

Zimmerman proposes a cyclical model of SRL based on social cognitive theory. This model has three phases for self-regulation: (1) forethought, which is the goal setting and planning before the assignment/study; (2) performance, which is when the students use various strategies, monitoring and controlling their learning; and (3) self-reflection, reflecting about the learning process after assignment/study [20, 21, 17].

Self-regulated learners are active participants in their learning and develop academic skills [22], adopting various learning strategies [21] during an academic assignment. SRL allows students to get acquainted with effective practices/strategies for their study, such as: time management; resource management; environmental management; incorporating feedback; and management of learning objectives and

results [22, 23]. Students construct their own meanings, goals, and strategies from the information available in the external environment and in their own minds [19].

SRL strategies (SRLS) are specific skills that are part of the SRL process, and can be taught for students to apply in real contexts [20, 21], such as: strategies for goal setting and planning, organizing and transforming, seeking information, rehearsing and memorizing, environmental strategies/structuring, seeking social assistance, self-consequences, records and monitoring, reviewing records, and self-evaluation [21].

The adoption of SRLs helps students obtain and retain knowledge about the adoption of a methodological approach and structured their learning, affecting the results of student learning [11]. According to [24], the application of SRLS are usually predictors of a good academic performance.

When students make effective self-reflection, they analyze how they learned, understood the objectives of the learning process and what is necessary to create conditions for success [18]. Also, they manage their learning and their commitment to meet challenges [18]. The interaction between the compromise, self-control, autonomy and students' self-discipline allows regulating their actions to achieve their learning goals [25].

The pedagogical context contributed for the learners engagement and resolve to achieve learning outcomes [26]. According to Wang et al. [17], in higher education it is important prepare students for the challenges of real work, and also to provide students with opportunities to develop their self-regulation and co-regulation skills, through activities that improve collaborative and active learning.

In engineering education, learning approaches are typically not aligned with the requirements of the labor market [27, 28], not giving priority to skills aligned with professional realities, such as active learning or integrating knowledge [29].

Students are immersed in business-like tasks mediated by structured and semi structured social interactions. Pedagogical techniques such as role playing stimulate students to learn about similar real-world situations, with problem-solving, active learning, providing opportunities to learn by doing and feedback for building new knowledge [30]. They also help develop professional identities [31].

3 The SimProgramming Approach: Immersive Features that Stimulate Self-regulated Learning

Pedrosa et al., including the authors of this paper, developed a teaching approach to help students learn computer programming in the transition from entry-level to advanced computer programming: the SimProgramming approach [16]. This approach is based on four conceptual foundations (ibid.): (1) business-like learning environment, (2) SRL; (3) co-regulation learning, and (4) formative assessment. Through these conceptual foundations, teaching strategies are adopted to stimulate SRLS by students with specific environment, roles, tasks, and deadlines during a course-long assignment.

The first conceptual foundation, *Business-like learning environment,* stipulates the simulation of a business-like environment, in order for students to have contact with aspects of their professional reality and teamwork expectations. Each participant plays a role and becomes immersed in the skills they have to develop during the assignment.

Problem-Based Learning (PBL) is used to promote collaborative discovery for the resolution of the problem [32].

The course lecturer plays the role of general manager, taking responsibility for the course content and monitoring. Course tutors or teaching assistants play 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]. Students play different roles as members of development teams.

Each team of students divides the work according to the role played by each member. For example: one student acts as team leader and the remaining students handle subsets of work (work packages). The team leader facilitates the integration of information and guides the group [6], making sure that team members keep a global view of the project context and status, integrating knowledge. Others students have each a specific role in the team, having to master their individual packages and cooperate with the team leader.

In the *conceptual foundation 2: SRL,* also detailed in [16], the goal is to promote students' SRLS through active participation and engagement in meaningful activities before, during, and after completion of academic work [25].

In SimProgramming, students are expected to be immersed: team members should focus the development of their role-specific skills, on research and exploration tasks for development of assigned problem/packages, throughout promoting active learning and helping students improve their self-regulation skills.

Each student has to solve their individual packages and contribute to the overall perspective of the team problem. The team leader integrates research and exploration output of all members, reporting weekly at project management meetings. He/she also ensures the information flow within the team. Weekly, each student makes a self-reflection about their work, ponders on what to do the following week, and reflects upon the factors that prevented him/her from achieving the team and the individual objectives.

Other aspects of immersion are time management and procrastination. It is common, in real work environments, that programming teams have to adapt their plans and overcome difficulties to meet deadlines for tasks. So, in SimProgramming we encourage students to develop the concept of having to do their work regularly and adopt study routines, by creating a context where tasks are performed continuously, with feedback and monitoring support for self-reflection and self-regulation.

SimProgramming also encourages *co-regulated learning* (*conceptual foundation 3*). Assignments include team tasks, namely: reports and presentations about the work.

The search for help among professional communities is also a common practice in real-world labour, so we encourage students to be involved in pre-existing online communities of professionals (outside academia) not just to seek help, but to help others, contributing to problem-solving and discussing the technologies under study or used in their future profession [3]. On this regard, during contact with tutors (in meetings, classes, and on-line), the goal is to stimulate students' initiative to search for social help (peers, teachers, tutors, etc.), not only clarify their doubts and difficulties. The tutors/assistants and the professor provide this support by advising on methods of gradual participation and involvement in communities, including suggestion of specific

tasks for clarification, and development the homogenous peer-based contributions and discussion, supporting community development, informal interactions and debate, which all were promoted and monitored via a Facebook group for the course.

Finally, is it well-known that companies conduct assessments of team performance. *Conceptual foundation 4: Formative Assessment*, aims to improve formative assessment throughout management feedback (tutors/assistants and professor).

SimProgramming Phases	Phase 1	Phase 2	Phase 3	Phase 4
Assignment Goals	-Searching for information on the technologies under study; -Interaction in online communities; -Group work: Initiate problem-solving.	- Integration of technologies; - Group work hands-on examples.	- Improving the assignment; - Final presentation with problem-solving	- Final improvement of the assignment.
Specifics Tasks	Weekly forms (individual) Weekly meetings between team leaders and tutors Report interaction in community of practice (team) Report about learning progress (team) Presentation of the team work	Weekly forms (individual) Weekly meetings between team leaders and tutors Report interaction in community of practice (team) Report about learning progress (team) Presentation of the team work	Weekly forms (individual) Weekly meetings between team leaders and tutors Report about learning progress (team) Presentation of the team work	Weekly forms (individual) Weekly meetings between team leaders and tutors Report about learning progress (team) Final Report (team) Grids about self-assessment of individual students and hetero-assessment by team members (of individuals) Extra task for extra credit or replacement credit (individuals or team)
Duration of the Phases	3 weeks	3 weeks	2 weeks	2 weeks

Fig. 1. SimProgramming phases: goals, specifics tasks and duration.

The Professor and assistants/tutors employ face-to-face and online contact to provide monitoring, meetings, and social media interactions, including motivational mentoring and feedback on individual package status. SimProgramming stipulates self-assessment of individual students and hetero-assessment by team members at the end.

3.1 SimProgramming Phases: Learning Assignment Process

In the SimProgramming approach [16], the learning assignment is developed along four phases and students have specific tasks in each phase (Fig. 1), based on the SimProgramming conceptual foundations presented above. During all phases, weekly meetings take place between tutors and team leaders, providing feedback for motivation, self-regulation, possible support for technical doubts, and internal team issues.

What are the individual weekly forms?

The individual weekly forms is where each student self-reflects upon his/her work, ponder on what to do the following week, and reflect upon the factors that prevented him/her or the team from achieving objectives [16]. Students need to answer 3 questions: (1) "What have you made this week for the assignment?"; (2) "What will you do next week for the assignment?"; and (3) "Any reason(s) for not completing tasks?".

4 Teaching Context and Learning Assignment

4.1 Teaching Context

Before reaching the Programming Methods 3 course (PM3, 2nd curricular year), students learned introductory programming in two previous courses, plus extra concepts in a Computational Logic course. PM3 is provided in parallel (joint lectures, but separate hands-on lessons) for students of two programmes of studies IE and ICT.

The goal in PM3 is to introduce the students to large-scale programming concepts, one of the learning objectives of the ACM/IEEE Computer Science Curricula (CSC). Specifically, students are introduced to the MVC architectural style, which divides programs among three blocks: the model (e.g., program state), the view (e.g., output), and the controller (e.g., program flow). The original MVC style proposal of Krasner and Pope [34], which handles input in the controller, is contrasted [4] with a more recent flavour proposed by Curry and Grace [7], which handles input in the view.

4.2 Learning Assignment in PM3

We combined face to face teaching techniques and technology-enhanced learning (TEL) [35] for support during the assignment. The tutors scheduled face-to-face meetings with team leaders, either individually or as a team, when they identified problems or difficulty fulfilling the tasks.

We used the Moodle LMS as the on-line environment for the professor and the tutors to track the development of the assignment, and organized the tasks into modules over several weeks. In the LMS, we provided supporting materials for development of

tasks, scheduling, overall objectives of the assignment and individual objectives of each task, a forum for doubts and for contacting tutors, and other course materials (e.g. slideshows used in lectures). Also, we employed other on-line tools to support students: e-mail, instant messaging (GTalk), Facebook, and a locally-developed course management system, SIDE [36] for students to submit their completed tasks.

The learning assignment is based on PBL [32]. 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 must develop a written document with a detailed explanation of the coding approaches they used to apply an MVC related architectural style to specific frameworks, libraries, and/or APIs [3, 4, 16, 37].

The SimProgramming approach was used throughout, along all the 4 phases, during 10 weeks of the academic semester, described ahead. In the 2012/2013 academic year, students formed 15 teams (Table 1). 11 teams successfully achieved the learning goals, two teams completed the requested tasks albeit falling short of achieving the goals, and two teams never actually started. Of the 97 students, 66 attained a final grade [16].

Table 1. Nr. of the students in assignment

Teams	Nr. students with a final grade	Comments
A	6/6	–
B	7/7	–
C	6/7	One student quitted the assignment
D	4/7	Of the 7 students, 3 quitted the assignment
E	6/7	One student quitted the assignment.
F	4/4	–
G	0/6	All students quitted the assignment, without even starting
H	7/7	–
I	5/6	One student quitted the assignment.
J	4/6	Two students never delivered the tasks, quitting the assignment
K	0/6	All students quitted the assignment, without even starting
L	2/7	Only 2 students delivered some of the requested tasks, performing the extra task. The remaining 5 members of the team never handed in any task (quitted at the beginning)
M	7/7	-
N	3/8	Only 3 were devoted and accomplished the tasks (the remaining 5 students quitted during the assignment)
O	4/5	One student quit the assignment
Total	66/97	

5 Methodology and Data Collection

During the 10 weeks of the assignment, each student had to submit their individual weekly form, with the exception of week 2, and week 10 (the final week). The delivery of the weekly forms changed along the SimProgramming phases (Table 2).

Table 2. Distribution of the weekly forms delivered in the SimProgramming phases

SimProgramming phases	Weeks	Nr. weekly forms delivered	Total
Phase 1	Week 1	81	150
	Week 3	69	
Phase 2	Week 4	55	168
	Week 5	65	
	Week 6	51	
Phase 3	Week 7	31	70
	Week 8	39	
Phase 4	Week 9	7	13
	Week 10 (in team)	6	
Total			401

As mentioned above, of the 97 students initially registering for the assignment, 66 completed phase 1, performing the tasks, and 31 others quit. In first week, 81 students delivered the weekly forms, but by week 3 this had decreased, and only 69 weekly forms were delivered. In Phase 2 and Phase 3 occasionally some students would miss a weekly delivery of forms. Finally, in phase 4, 7 students delivered their weekly forms on week 9, and 6 of the 15 teams delivered the team form.

As Table 2 shows, we observed stability during the initial weeks. However, on the week 7 there was a sharp decline. This was the week after Easter break, and students reported being a time when they had many mid-terms and assignment deadlines piling up:

> *"I had mid-terms and works deadlines for other courses, and I feel really tired, since we are near the end of the middle of the second semester."* (E38, Week 7, 22/03/2013)

> *"Although there are no classes during the Easter break, the work remained the same. Now, what's starting to worry me are the final assignments, mainly from courses on [Another course] and [Yet another course]."* (E13, Week 7, 29/03/2013)

> "*Lack of time was the main cause of the failures that occurred in the presentation. It is not easy to manage and bring together a group of six elements: we all have different courses and assignments, and this is sometimes also an impediment to reaching the goal of the group (…)"* (E5, Week 8, 12/04/2013)

In this paper, we analyse the 401 weekly forms using thematic analysis [38] with the goal of identifying the self-regulated learning strategies mentioned by students during the assignment. We constructed content analysis matrices based on background about the types of self-regulated learning strategies, identifying difficulties and factors that they believed influenced their motivation.

We organized content into categories, subcategories, indicators, and recording units (snippet sentences), which were restated during the process of content analysis. Then, we conducted a cyclical process of improvement, synthesis, and reflection. The steps adopted for the data analyses were as follows:

1. Construction of content analysis matrices for each team, with the SRLS reported by students (phrases/snippet sentences that students reported on the weekly forms,

explaining what they did). The content analysis matrices are composed of grid lines (each line for a strategy –the "indicators"); and columns to record in which week it was reported by students. In the cells we entered codes identifying the student reporting that strategy that week (e.g. E.3).
2. Afterwards, we developed general syntheses of each team references (students) for each of the indicators.
3. For each subcategory of the strategies (e.g. *Organizing and planning strategies*) we counted the number of students who reported each indicator (e.g. 1.1 = 113).
4. Finally, we did a general syntheses of the indicators in each of the phases.

6 Results and Discussion

6.1 Self-regulated Learning Strategies – Results of Analysis of the Weekly Forms

Regarding organizing and planning strategies for the assignment, detailed in Table 3, during the early stages (Phase 1 and Phase 2) the strategies most commonly adopted by students were information search, checking the material provided by the tutors/professor or other courses, recording of practices in online communities and team meeting to define tasks.

Table 3. Organizing and planning strategies

Indicators	SimProgramming phases			
	Phase 1 (N = 150)	Phase 2 (N = 168)	Phase 3 (N = 70)	Phase 4 (N = 13)
3.1. Organizing – Information search	113	103	17	1
3.2. Organizing – Collected information	23	43	23	1
3.3. Planning – Work plan development	2	2	0	0
3.4. Planning - Following guidelines provided by tutors and professor	1	4	2	0
3.5. Had no planned strategy	0	3	0	0
3.6. Transforming – Drafting notes about collected information	9	3	2	0
3.7. Transforming - Application of existing knowledge about the practice	16	74	38	13
3.8. Organizing - Understand the project goals	4	0	1	0
3.9. Organizing - Checking the material provided by the teacher or other courses	16	5	0	0
3.10. Transforming - Understanding (learning) through the collected information search	53	82	22	4
3.11. Organizing - Recording practices online communities	66	15	0	0
3.12. Planning - Team meeting to define tasks	98	89	37	5
3.13. Planning - Meeting scheduling with tutors	1	2	0	0
3.14. Organizing - Meeting schedule with team colleagues	4	3	1	0
3.15. Organizing - Meeting with tutors	4	3	1	0
3.16. Planning - Defining specific tasks for next week	33	55	33	4

Table 4. Identifying of the difficulties in time management

Indicators	SimProgramming phases			
	Phase 1 (N = 150)	Phase 2 (N = 168)	Phase 3 (N = 70)	Phase 4 (N = 13)
4.1. Time management (TM) - Lack of time	2	3	1	0
4.2. TM- Lack of time due to other responsibilities	5	11	5	0
4.3. TM- Initiating the activity at the last moment (procrastination)	1	6	2	0
4.4. TM- Submitted in following week	8	10	4	0
4.5. TM - Difficulties in TM due to work in others courses or tests	49	80	30	1

Afterwards, still in phase 2, students initiate the application of existing knowledge about the practice, understanding (learning) through the collected information search, and defining specific tasks for the following week. In phase 3, the strategies remained the same as those applied in the preceding phase.

In the end (phase 4), students deepen their practical skills in teamwork (indicators 1.7 and 1.12). During assignment execution, other strategies were mentioned less often.

As shown on Table 4, the lack of time and difficulties in time management were frequently mentioned by students, due to a diversity of responsibilities but mainly because many tasks and tests that had to be performed in different courses. The most critical phases were phase 2 and phase 3.

As shown in Table 5, the difficulties students encountered while performing the assignment were at the level of theoretical content and practical implementation of the assignment. Difficulties in team work and scarce feedback obtained from on-line communities were also mentioned, mainly in phase 2. In phase 3 the most mentioned difficulties were about the implementation of the practical component.

Strategies mentioned by students to resolve their difficulties, as shown in Table 6, were varied, with a prevalence of SOA (teachers, peers, others). Only in the early phases (phase 1 and phase 2) did the students mention interaction with online communities.

Table 5. Identifying difficulties in the assignment

Indicators	SimProgramming phases			
	Phase 1 (N = 150)	Phase 2 (N = 168)	Phase 3 (N = 70)	Phase 4 (N = 13)
5.1. Difficulties – Theoretical knowledge about the technology being studied	3	16	0	0
5.2. Difficulties – The practical component implementation	2	19	14	1
5.3. Difficulties – Team work	9	15	3	1
5.4. Difficulties – Too many tasks per week	8	1	0	0
5.5. Difficulties - Scarce feedback obtained in on-line community	11	41	9	1

Table 6. Strategies for resolution of difficulties

Indicators	SimProgramming phases			
	Phase 1 (N = 150)	Phase 2 (N = 168)	Phase 3 (N = 70)	Phase 4 (N = 13)
6.1. Resolution of Difficulties (RD) - Use of practical exercises	0	6	1	0
6.2. Seeking Social Assistance (SOA) – Teachers	3	7	6	4
6.3. SOA – Team peers	2	3	0	0
6.4. SOA - Senior colleagues	4	1	1	1
6.5. SOA - Others	0	4	1	2
6.6. RD - Interactions in online communities	34	32	7	0

Also, some students expressed factors that affected their motivation during the assignment, as shown on Table 7. Most are of personal nature, for example, the need to achieve success in PM3 in order to attain completion of the programme of studies; the will to learn; interest in programming; the grade impact of the assignment. But some factors are linked to interpersonal and social dimensions, namely teamwork, being the leader with the associated responsibility, and the feedback obtained.

Table 7. Factors influencing motivation

Indicators	SimProgramming phases			
	Phase 1 (N = 150)	Phase 2 (N = 168)	Phase 3 (N = 70)	Phase 4 (N = 13)
7.1. Motivation (MT) – Completing PM3	0	3	0	0
7.2. MT - Interest in programming	1	0	0	0
7.3. MT- Comply with an obligation	0	3	0	0
7.4. MT - Grade impact of the assignment	2	2	0	0
7.5. MT - Responsibility for teamwork	1	3	0	0
7.6. MT – Found the process interesting (SimProgramming approach)	0	1	0	0
7.7. MT – Learning	3	2	0	0
7.8. Lack of motivation – Is tired (of studying)	0	0	0	1
7.9. Lack of motivation – Overall grades are not enough to complete the course	0	1	0	0
7.10. MT – Feedback obtained with tutors	0	6	0	0

Regarding self-reflection by students about their completed tasks, detailed in Table 8, few students made a thorough self-reflection with details about their performance in the required tasks and self-learning. They generically referred only on having achieved or not their goals. Only in the end phase did the students become more reflective.

Table 8. Self-reflection

Indicators	SimProgramming phases			
	Phase 1 (N = 150)	Phase 2 (N = 168)	Phase 3 (N = 70)	Phase 4 (N = 13)
8.1. Self-reflection (SR)- Achieved the goals	68	57	37	3
8.2. SR - Achieved the goals with difficulties	16	31	10	1
8.3. SR - Aware of lacking team leader skills - asked to be replaced	1	0	0	0
8.4. SR- Wants to get feedback from tutors to know if the objectives were achieved	1	0	0	0
8.5. SR - Did not reach the goals due to overload	27	44	20	6
8.6. SR - Reflection on specific tasks	9	24	13	8

7 Conclusions

Along the phases of the SimProgramming approach, the students have shown in their weekly forms that they were adopting many different strategies in each phase. In the early phases (phase 1 and phase 2), strategies were mostly about organization and planning. In the following stages (part of phase 2, but mostly phases 3 and 4) this shifted towards the application and transformation of information: application of theoretical knowledge and implementation of the hands-on component (programming). These strategies are skills necessary for the development of project teamwork in real-world labour. Students improved their competence in such skills during the SimProgramming phases and are expectably better prepared for the transition to the real-world labour.

In the weekly forms, students mentioned SRLS, especially on organizing and planning. They also mentioned strategies for resolution of difficulties, identifying difficulties in time management and difficulties in assignment, and factors influencing their motivation – strategies that had also been identified in our earlier work [37]. This highlights the difficulties students feel managing their time because of tasks and tests they need to account for in the various courses throughout the semester.

Some students mentioned in the weekly forms the adoption of transformation strategies and showed that they were aware of their specific difficulties in the tasks, aspects that were not reported so often in our previous work [37].

The students engaged in self-reflection about their learning, explaining whether or not they had reached their personal goals or the goals of the SimProgramming approach. However, only some students did a more thorough self-reflection about their performance. This confirms the need to help students become aware of the strategies that they can take to improve self-regulated learning [37]. Interestingly, as a team, the students reflected with significant detail about their performance.

We believe that the weekly forms or a similar instrument (for example, weekly meetings with students and tutors) contribute to the improvement of the adoption of self-regulated learning strategies because they raised students' awareness about important skills/strategies for real-world labour.

Acknowledgments. Pedrosa, D. wishes to thank the Fundação para a Ciência e Tecnologia (FCT), Portugal, for Ph.D. Grant SFRH/BD/87815/2012.

This work was partly financed by the ERDF – European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme within project «POCI-01-0145-FEDER-006961», and by National Funds through the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) as part of project «UID/EEA/50014/2013».

We would like to thank all the students and teachers who collaborated on this research.

References

1. Lahtinen, E., Ala-Mutka, K., Järvinen, H.M.: A study of the difficulties of novice programmers. ACM SIGCSE Bull. **37**(3), 14–18 (2005)
2. Gomes, A., Mendes, A.J.: Learning to program-difficulties and solutions. In: International Conference on Engineering Education–ICEE, Coimbra, Portugal (2007)
3. Morgado, L., Fonseca, B., Martins, P., Paredes, H., Cruz, G., Maia, A.M., Nunes, R., Santos, A.: 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), pp. 1–7. IEEE (2012)
4. Nunes, R.R., Pedrosa, D., Fonseca, B., Paredes, H., Cravino, J., Morgado, L., Martins, P.: Enhancing students' motivation to learn software engineering programming techniques: a collaborative and social interaction approach. In: Antona, M., Stephanidis, C. (eds.) UAHCI 2015. LNCS, vol. 9177, pp. 189–201. Springer, Heidelberg (2015)
5. Kumar, B.: Gamification in education-learn computer programming with fun. Int. J. Comput. Distrib. Syst. **2**(1), 46–53 (2012)
6. Sancho, P., Moreno-Ger, P., Fuentes-Fernández, R., Fernández-Manjón, B.: Adaptive role playing games: an immersive approach for problem based learning. Educ. Technol. Soc. **12** (4), 110–124 (2009)
7. Curry, E., Grace, P.: Flexible self-management using the model-view-controller pattern. IEEE Softw. **25**(3), 84–90 (2008)
8. Cagiltay, N.E.: Teaching software engineering by means of computer-game development: challenges and opportunities. BJET **38**(3), 405–415 (2007)
9. Jenkins, T.: On the difficulty of learning to program. In: Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences, vol. 4, pp. 53–58 (2002)
10. Fernández, E., Bernardo, A., Suárez, N., Cerezo, R., Núñez, J.C., Rosário, P.: Predicción del uso de estrategias de autorregulación en educación superior. Anales de Psicología **29**(3), 865–875 (2013)
11. Broadbent, J., Poon, W.L.: Self-regulated learning strategies & academic achievement in online higher education learning environments: a systematic review. Internet High. Educ. **27**, 1–13 (2015)
12. Zimmerman, B.J.: Investigating self-regulation and motivation: historical background, methodological developments, and future prospects. Am. Educ. Res. J. **45**(1), 166–183 (2008)
13. Cazan, A.M.: Teaching self regulated learning strategies for psychology students. Procedia-Soc. Behav. Sci. **78**, 743–747 (2013)

14. Bergin, S., Ronan R., Desmond, T.: Examining the role of self-regulated learning on introductory programming performance. In: Proceedings of the First International Workshop on Computing Education Research. ACM (2005)
15. Alharbi, A., Paul, D., Henskens, F., Hannaford, M.: An investigation into the learning styles and self-regulated learning strategies for computer science students. In: Proceedings of Ascilite (2011)
16. Pedrosa, D., Cravino, J., Morgado, L., Barreira, C., Nunes, R.R., Martins, P., Paredes, H.: Simprogramming: the development of an integrated teaching approach for computer programming in higher education. To Appear in Proceedings 10th Annual International Technology, Education and Development Conference (INTED 2016), Valencia, Spain (2016)
17. Räisänen, M., Postareff, L., Lindblom-Ylänne, S.: University students' self- and co-regulation of learning and processes of understanding: a person oriented approach. Learn. Individ. Differ. (2016). http://dx.doi.org/10.1016/j.lindif.2016.01.006
18. Hadwin, A.F., Järvelä, S., Miller, M.: Self-regulated, co-regulated, and socially shared regulation of learning. In: Zimmerman, B.J., Schunk, D.H. (eds.) Handbook of Selfregulation of Learning and Performance, pp. 65–84. Routledge, New York (2011)
19. Pintrich, P.R.: A conceptual framework for assessing motivation and self-regulated learning in college students. Educ. Psychol. Rev. **16**(4), 385–407 (2004)
20. Zimmerman, B.J., Schunk, D.H.: Self-regulated learning and performance. An introduction and an overview. In: Zimmerman, B.J., Schunk, D.H. (eds.) Handbook of Self-regulation of Learning and Performance, pp. 1–12. Routledge, New York (2011)
21. Zimmerman, B.J.: From cognitive modeling to self-regulation: a social cognitive career path. Educ. Psychol. **48**(3), 135–147 (2013)
22. Clark, I.: Formative assessment: assessment is for self-regulated learning. Educa. Psychol. Rev. **24**(2), 205–249 (2012)
23. Nicol, D.J., Macfarlane-Dick, D.: Formative assessment and self-regulated learning: a model and seven principles of good feedback practice. Stud. High. Educ. **31**(2), 199–218 (2006)
24. Wang, C.H., Shannon, D., Ross, M.: Students' characteristics, self-regulated learning, technology self-efficacy, and course outcomes in online learning. Distance Educ. **34**(3), 302–323 (2013)
25. Hattie, J., Timperley, H.: The power of feedback. Rev. Educ. Res. **77**(1), 81–112 (2007)
26. Johri, A., Olds, B.M.: Situated engineering learning: bridging engineering education research and the learning sciences. J. Eng. Educ. **100**(1), 151–185 (2011)
27. Duderstadt, J.J.: Engineering for a changing world. In: Grasso, D., Burkins, M.B. (eds.) Holistic Engineering Education, pp. 17–35. Springer, New York (2010)
28. Sheppard, S.D., Macatangay, K., Colby, A., Sullivan, W.M.: Educating Engineers: Designing for the Future of the Field, vol. 2. Jossey-Bass, San Francisco (2008)
29. Adams, R., Evangelou, D., English, L., De Figueiredo, A.D., Mousoulides, N., Pawley, A. L., Schiefellite, C., Stevens, R., Svinicki, M., Trenor, J.M., Wilson, D.M.: Multiple perspectives on engaging future engineers. J. Eng. Educ. **100**, 48–88 (2011)
30. Bransford, J., Brown, A., Cocking, R. (eds.): How People Learn: Brain, Mind, Experience, and School, Committee on Developments in the Science of Learning, Commission on Behavioral and Social Sciences and Education. NRC, National Academy Press, Washington, D.C. (2000)
31. Duarte, M.O., Oliveira, I., Félix, H., Carrilho, D., Pereira, A., Direito, I.: Active classrooms: role-playing experience in telecommunications engineering education. Int. J. Eng. Educ. **27** (3), 604–609 (2011)

32. Savery, J.R.: 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 (2015)
33. Schwaber, K.: Agile Project Management with Scrum. Microsoft Press, Redmond (2004)
34. Krasner, G., Pope, S.: A description of the model view controller paradigm in the small-talk-80 system. J. Object Oriented Program. **1**(3), 26–49 (1988)
35. Kirkwood, A., Price, L.: Technology-enhanced learning and teaching in higher education: what is 'enhanced' and how do we know? A critical literature review. Learn. Media Technol. **39**(1), 6–36 (2014)
36. Barbosa, L., Alves, P., Barroso, J.: SIDE - teaching support information system. In: 6th Iberian Conference on Information Systems and Technologies (CISTI), pp. 1–6. IEEE (2011)
37. Pedrosa, D., Cravino, J., Morgado, L., Barreira, C.: Self-regulated learning in higher education: strategies adopted by computer programming students. To Appear in Proceedings of 8th International Symposium on Project Approaches in Engineering Education (PAEE), Guimarães, Portugal (2016b, in press)
38. Braun, V., Clarke, V.: Using thematic analysis in psychology. Qual. Res. Psychol. **3**(2), 77–101 (2006)