





Computer Supported Collaborative Learning for Programming: A Systematic Review

Ricardo Sol¹^a, Elci Alcione Santos²^b, Manuel C. Reis³^c and Lucas Pereira¹^d

¹ITI, LARSyS, Polo Científico e Tecnológico da Madeira, floor-2, 9020-105 Madeira, Portugal

²Faculty of Exact Sciences and Engineering, University of Madeira, Madeira, Portugal

³Department of Engineering, University of Trás-os-Montes e Alto Douro, Vila Real, Portugal

Keywords: Computer Supported Collaborative Learning, Programming, Educational Technology, Reviews.


Abstract: The objective of this paper is to present the current evidence relative to the effectiveness of computer supported collaborative learning as a pedagogical tool in teaching programming. A systematic literature review in the IEEE Xplore, Web of Science, and ACM Digital Libraries was performed with studies that investigated factors affecting the effectiveness of computer supported collaborative learning for students learning programming and studies that measured the effectiveness of computer supported collaborative learning for students learning programming. Twelve papers were used in the analysis. The results showed that the object oriented programming languages are the ones that have been most frequently adopted by educators who use computer-supported collaborative learning as tools to teach programming, that course critique surveys and questionnaires are the most frequently reported methods used to assess the effectiveness of computer-supported collaborative learning interventions, and that the amount of participants who have taken part in research to evaluate the value of computer-supported collaborative learning in teaching programming varies notably between studies. Finally, in total, 83.3% of the included papers report that computer supported collaborative learning is an effective teaching tool and can help programmers in their studies.


1 INTRODUCTION


Learning programming can be a difficult task. Many references can be found in literature, all over the world, pertaining to the difficulties numerous students have in understanding and learning programming courses. A miscellaneous collection of reasons has been identified as the difficulties demonstrated by these students. Some authors highlight that a preexisting mental model of knowledge can affect the acquisition and use of programming concepts. The literature identified numerous bugs that are made by learners who can show a sort of negative influence from natural language or rudimentary models of how a process works (Gray et al., 1993).


Customarily, the teaching of initial programming has highlighted the writing of programs from the

outset. The daily analysis of matters is clearly planned in relation to the technology, not the cognitive growth of the learner. These methods start from the fifth and sixth stages of Bloom's Taxonomy of Educational Objective, when these last two stages are contingent upon proficiency in the previous four stages (Lister, 2000). The programming task crosses the Learning Style Inventory environments (Kolb, 1985), depending on whether a learner is trying to solve a problem (a symbolically complex environment), employing abilities (behaviorally complex), recognizing and understanding the association between notions (perceptually complex). This may imply that diverse learning styles come to the fore throughout the whole programming procedure (Byrne and Lyons, 2000). Constructivism applied to programming in practice has certainly not obeyed theoretical foundations. Instead of an improvised

^a  <https://orcid.org/0000-0003-4333-7140>

^b  <https://orcid.org/0000-0002-1189-4076>

^c  <https://orcid.org/0000-0002-8872-5721>

^d  <https://orcid.org/0000-0002-9110-8775>

attitude to planning and applying transformation, a strict research outline is needed in order to improve transformations grounded on meaningful theoretical understanding (Bruce and McMahon, 2002). Focused upon the ability of learners to consistently execute code are aspects of programming on which educators should assess learners. Also, to emphasize exclusively upon those parts of programming is to emphasize upon the three inferior stages of the SOLO taxonomy (Collis & Biggs, 1979): the pre-structural, unistructural, and multi-structural levels. From think-aloud responses, the researchers found that teachers tended to show a SOLO relational response on minor reading issues, while learners leaned towards showing a multi-structural response (Lister et al., 2006). It was found that there are at least two cognitive factors that show themselves as opportunities that may make learning of programming problematic, being those of learning style and of motivation (Jenkins, 2002).

The results of a survey about programming concepts presented to 500 learners worldwide confirm that the most difficult concepts to learn are the ones that need understanding of greater entities of the program as opposed to just details. The results also sustained the notions that abstract concepts like pointers and memory handling are hard to learn. The results also exposed a set of topics (e.g., language libraries, input and output) that would perhaps need additional attention, since understand them was not related to understand the essentials of programming (Lahtinen et al., 2005). An empirical study was motivated by the idea that diverse people create diverse outlines of information in any new learning process and proven that how each learner deals with problems in a singular way is grounded on their mental model. The preliminary study implies that accomplishment in the first phase of an introductory programming course is anticipated, by observing steadiness in use of mental models that learners apply to a initial programming problem even earlier than they have had any interaction with programming (Dehnadi, 2006).

Nevertheless, the biggest problem of beginner programmers does not seem to be the understanding of basic concepts but instead learning to use them. However, research shows that learners of programming operating collaboratively beat solo programmers (Nosek, 1998). Over a qualitative and quantitative analysis, collaborative practices showed encouraging results on fundamental characteristics of learning programming skills, i.e., learning and motivation. Indeed, as far as learning is concerned, the quantitative analysis showed that it outperformed

solo programming, because the programmers inserted less code anomalies (Estácio et al., 2015).

The results of several studies do not back the impression that cognitive styles are fixed traits and give more support for the plasticity of cognitive styles, in those occasions where learners are helped by computer-supported systems (Angeli et al., 2016).

An intelligent tutoring is a computer system that targets to offer instruction or feedback to students and is more successful than the customary classes, learning is faster and foments better performance on tests (Reiser et al., 1985).

We hope that this article will make clear which claims of computer supported collaborative learning are supported by scientific studies. We aim to present the prevalence of these claims within a systematic sample. Specifically, the objective of the review is to answer five research questions stated in the methodology.

The method that has been implemented in this Systematic Literature Review is described in depth in Section 2 where the research questions are presented, while Section 3 is dedicated to the results of the literature review search. In Section 4, a discussion takes place in an attempt to answer the five research questions and in regard to different aspects of the literature review. This is followed by a conclusion from the literature review in Section 5.

2 METHODOLOGY

2.1 Research Questions

This literature review is influenced by the work of Kitchenham and Charters (2010) that proposed guidelines for performing Systematic Literature Reviews in Software Engineering. An initial protocol was developed as part of this literature review. The primary focus of this literature review is to understand and identify computer-supported systems for collaboratively learning for programming. The following research questions were formulated in order to achieve this goal:

1. What computer languages are being taught?
2. What are the characteristics of the learners being taught?
3. What types of research studies are performed to investigate the computer supported collaborative learning?
4. What is the number of participants in studies that are being performed by researchers?

5. Do studies suggest that using computer supported collaborative learning for programming is effective?

2.2 Search

Within the field there are several expressions that relate to programming education, collaborative learning and tutoring systems. It was used a Boolean search string that included synonyms:

("collaborative learning" OR "cooperative learning") AND ("intelligent tutor*" OR "adaptive tutor*" OR "cognitive tutor*" OR "smart tutor*") AND programming AND (novice OR beginner OR introductory OR teaching OR learning OR CS1 OR "first time").

The systematic sample was retrieved from the IEEE Xplore, Web of Science, and ACM Digital Libraries. Whenever a paper was found suitable, it was added to the list of papers qualified for the synthesis. Web of Science was the last to be looked at, and thus it only returned duplicate studies.

2.3 Inclusion and Exclusion Criteria

The inclusion and exclusion criteria were used to guaranty that only significant literature was added to the literature review. In order to determine whether articles met the inclusion or exclusion criteria abstracts were read.

2.3.1 Inclusion Criteria

1. Publications that have tutoring systems used by students learning programming collaboratively.
2. If papers reported the same study, only the latest was added.
3. Papers were added independently of their date of publications.
4. Relevant grey literature is accepted.

2.3.2 Exclusion Criteria

1. Publications that do not report a system.
2. When only the Abstract and not the full text is available.
3. Publications with Systems that are only partially prototyped.
4. Position papers, editorials, and letters were all excluded.

2.4 Study Quality Assessment

To aid the data extraction process, a form was created, which was used to collect evidence relating to the

research questions and measure the quality of the primary studies. When designing the quality checklist of the study, the eleven criteria discussed by Dyba and Dingsøy were used, (Dyba and Dingsøy, 2008) that were based in the Critical Appraisal Program (Gilb, 2005). The checklist was comprised of eleven general questions to measure the quality of both quantitative and qualitative studies according to the following ratio scale: Yes = 1 point, Partially = 0,5 point, and No = 0 points. Ranging the resulting total quality score for each study between 0 (very poor) and 11 (very good).

The eleven criteria used to assess the quality of each publication are quoted as follows:

1. Is the paper based on research or is it a 'lessons learned' report based on expert opinion?
2. Is there a clear statement of the aims of the research?
3. Is there an adequate description of the context in which the research was carried out?
4. Was the research design appropriate in order to address the aims of the research?
5. Was the recruitment strategy appropriate with regards to the aims of the research?
6. Was there a control group with which to compare?
7. Was the data collected in a way that addressed the research issue?
8. Was the data analysis sufficiently rigorous?
9. Has the relationship between researcher and participants been considered to an adequate degree?
10. Is there a clear statement of findings?
11. Is the study of value for research or practice?

The first two of these criteria represent the minimum quality threshold that was observed during this literature review. The following nine criteria are intended to determine the rigor and credibility of the research methods employed as well as the relevance of each paper in relation to the literature review.

2.5 Data Extraction

When publications were identified as meeting the criteria for inclusion, the full text was read. Then in order to answer the research questions the following data were extracted from each publication included in the literature review:

- Publication type;
- Publication aims and objectives;
- Methodology of the publication;
- Number of participants in a study;
- How data was gathered and analyzed during the study;
- Characteristics of the learners being tutored;

- Programming languages taught by the tutor;
- Did the system(s) include task related to learners generated program planning or visualizations;
- Did the system(s) use visualizations or plans as instructional resources?

One reviewer extracted all data during the first semester of 2020. In order to validate the extraction process, a random sample comprising of 20% of the total number of primary studies had their data extracted by a second reviewer. These results were then compared. Whenever the data extracted differed, where differences never surpassed more than 7%, such differences were discussed until consensus was reached. The data extraction strategy was deemed to be appropriate. All extracted data was stored in a spread sheet.

3 RESULTS

In this section the synthesis of the literature review is presented, beginning with the analysis from the literature search results. During the selection process, the IEEE Xplore, Web of Science, and ACM Digital Libraries were chosen as the baseline databases due to its reputation.

3.1 Search Results

The initial phase of the search process identified two hundred and four publications matching the search string. Of these, only thirty-seven were potentially relevant based on the screening of titles and abstracts. Each of these thirty-seven studies was filtered according to the exclusion and inclusion criteria before being accepted in the literature review list. If titles and abstracts were not sufficient to identify the relevance of a paper, full articles were read. It was also checked if there were any very similar studies or duplicate studies that were published in more than one publication.

Based on the search, 12 studies (32,4% of the 37 studies) were accepted in the literature review list after a detailed assessment of the abstract, full text, and exclusion of duplicates. In the following section, it is presented the quality assessment results are presented (see the Appendix for the list of studies used in this literature review).

3.2 Quality Assessment Results

Each study had been assigned a quality score out of eleven. Only two of the articles included in the list

were not based on research or presented a “lessons learned account”, but offered some description of the context in which the research was carried out. All articles clearly stated the aims of the research; however only one had an adequate relationship between participants and researchers.

More than half of the studies had an adequate recruitment strategy. None of the studies included in the literature review was awarded the maximum score of 11, with the highest score awarded being 9.5. The average quality score of publications included in the literature review was 8.29 with standard deviation of 1.2. The lowest score that articles were awarded was 1.5. Because the average quality score of the included publications varied, it was decided to maintain all papers due to the small number of publications selected. In the following section, we present the results for the literature review research questions.

3.3 Research Questions Results

Answers to the research questions outlined in Section 2.1 will now be discussed.

3.3.1 What Computer Languages Are Being Taught?

When analyzing the studies included in the literature review, three different categories were established regarding the programming languages used: Object Oriented, Non Object Oriented, and Dedicated. Object oriented languages were the largest contributor to the literature review having been the main programming language used in seven papers. Evidence was collected that stresses how efforts have been made to use designed programming languages in order to teach programming principles, as LeJOS [NOGUEZ07]. LeJOS is an open-source project created to develop a technological infrastructure to develop software to robots using Java technology. Evidence was also collected that stresses how efforts have been made to use web-programming languages in order to teach programming principles [STARBIRD11, WANG09].

3.3.2 What Are the Characteristics of the Learners Being Taught?

The diverse context of each study was scrutinized in order to determine the characteristics of the students that have been taught programming. Two different groups were established as a result of this and these were ‘University’, and ‘various’. Out of the 12 papers 8 reported on the use of technology in a university setting. Three discussed the implementation in

multiple environments [GUO15, JENKINS12, STARBIRD11].

3.3.3 What Types of Research Studies Are Performed to Investigate the Computer Supported Collaborative Learning?

The use of surveys and questionnaires was equally commonly found method by which the reviewed studies evaluated their findings and proposals (six papers reported the use of such methods). Log and video record also have been described (each in one paper). Analysis of student grades has also been reported in one paper that also examined the impact that computer-supported tools had upon retention rates [DING17]. In addition, comparative analysis has also taken place; this has included contrasting the effect on the learners of learning with computer-supported tools to learning without (one paper). Two papers included in the literature review were ‘lessons learned’ [JOVANOVIC15, LEUNG07].

3.3.4 What Is the Number of Participants of Studies That Are Being Performed by Researchers?

There are two papers included in the review that have examples of ‘lessons learned’ or experience style reports, whereas six papers offer evidence that an empirical study took place. The scale of studies included in the review varied notably. These ranged from small-scale studies that contained 6 participants through to larger studies that reported sample sizes with more than 600 participants. Ten papers report the exact number of participants that took part in the research performed. In contrast one paper discusses conducting experiments or collecting information from participants but did not state the precise number of participants involved [LEUNG07].

3.3.5 Do Studies Suggest That using Computer Supported Collaborative Learning for Programming Is Effective?

After analyzing the papers included in the literature review, it is possible to show an analysis on whether the included publications report the use of computer-supported collaborative learning (CSCL) to be an effective intervention in the learning of programming. Of the 12 papers included in the literature review, 10 papers report that the use of CSCL is effective when learning introductory programming concepts.

One paper was identified to be “unclassifiable” because it did not provide a measure of the effectiveness of CSCL when used in such context.

3.4 Limitations of the Review

The most important limitation of the validity of the literature review is the fact that it was performed only in IEEE Xplore, Web of Science, and ACM Digital Libraries.

Other important limitations of the validity of the literature review are in relation to bias in the selection of papers and imprecise data extracted. Search strings were devised as the literature review employed exclusively the electronic resources of IEEE Xplore, Web of Science, and ACM Digital Libraries. These were established after applying trial searches. Notwithstanding this, it is not possible to assure that all studies in the IEEE Xplore, Web of Science, and ACM Digital Libraries relevant to the topic under consideration were returned and there is a fair risk that some studies may have been omitted due to the search terms used. Furthermore, the phenomena where ‘negative’ results are less likely to be published, known as ‘publication bias’ may also have had a fair impact on the findings of the literature review, though it is difficult to determine whether this was the case. The data extraction procedure can also have been undesirably impacted by bias when choosing publications. This is due to the fact that data extraction procedure has been performed by only one reviewer. In addition, it is possible that the inclusion and exclusion criteria may have unintentionally disqualified some relevant publications. This is because the applied criteria stopped being of added papers that contained no ‘lessons learned’ element. Finally, non-English language and abstract only papers were excluded from addition to the literature review. However, no papers were found that were written in another language probably due to the search string. Furthermore, by excluding the publications where only the abstract of the paper is available, one could have unintentionally avoided acceptable publications from being included in the literature review.

4 DISCUSSION

In this section aspects of additional analysis that has been assumed to corroborate the results of the literature review are presented. Furthermore, a discussion regarding the findings of the literature review is also portrayed.

It was noted throughout this process that of the 12 papers included in the literature review 9 were published in Conference Proceedings, and 2 in Journals. Of the 9 papers published in conference proceedings 2 were short papers.

Several interpretations can be made as a result of the literature review. When looking at the original quality score, with the average being 8.29 out of 11, it is accepted that this number is satisfactory. A high proportion of publications contained in the initial set of 12 required fundamental experimental features like a control group, whereas the relationship between researcher and participants was often considered to be of a poor standard. This is due to 2 of the 12 papers included in the review being 'lessons learned' or experience workshop reports. Such papers do not score well in the quality assessment criteria that has been used.

Two large-scale comparative studies were included in the literature review. Only one paper reported a semester long experiment that compared the results of more than 600 participants. Usually, a large study may be considered to offer far more compelling evidence than the results of small non-comparative studies. However, two small studies describe the results of an experiment that compared the results from the participants on tests from both computer-supported collaborative learning and non computer-supported collaborative learning programming sessions.

Five research questions were created in order to determine the value of using CSCL when teaching programming. Several findings and tendencies, regarding the learning of programming using CSCL, can be noticed as a result.

These comprise the observations that:

- The object-oriented programming languages are the ones that have been most frequently adopted by educators.
- Course critique surveys and questionnaires are the most usually described methods used to assess the effectiveness of CSCL sessions.
- The amount of participants who have taken part in research to assess the value of CSCL in learning programming varies a lot between studies.

In general, the findings of the literature review imply that the use of CSCL can be an effective learning tool when used in a programming course. This is evident as 10 of the 12 papers included in the literature review clearly state that computer-supported collaborative learning is valuable when used in such a way.

5 CONCLUSIONS

This review has examined the effectiveness of using CSCL to teach programming by using a systematic literature review methodology. After employing a search strategy, 12 papers were initially included in the literature review. The findings of the literature review show how the use of CSCL can be an effective learning tool when used in programming courses. Definitely, 83.3% of the publications included in the review reported this.

Several findings and tendencies, with regards to the teaching of programming using CSCL, have been noticed as a result of the literature review. These comprise the detection firstly of object-oriented programming languages as the ones that have been most frequently adopted by educators who use CSCL as tools to teach programming. Secondly, that course critique surveys and questionnaires are the most frequently reported methods used to assess the effectiveness of CSCL interventions. Thirdly, that the amount of participants who have taken part in research to evaluate the value of CSCL in teaching programming varies notably between studies.

The most significant finding of the literature review, which researchers should have in account, nevertheless, is that there is an obvious need for large-scale and high-quality research to be undertaken in order to discover the true effectiveness of CSCL as a programming teaching tool.

Due to the fact that the included publications utilize a broad variety of methods to collect data, in combination with the samples size, statistical analysis methods have not been used during this study and so these results are not statistically significant. As a consequence, additional research is needed in order to establish the true effectiveness of CSCL that can be used to support the teaching of programming. However, this work emphasizes that there is a lot of potential for future work of researchers within the field to build upon the body of existing knowledge documented in the literature review.

Upon completion of this first study, the identification of relevant literature will continue with the second search phase. During the second phase, all of the references in the papers identified in the first phase will be reviewed.

This systematic literature review shows that there is a clear need for large-scale and high-quality research to be done in order to determine the true effectiveness of computer supported collaborative learning as a programming teaching tool. From this study, it is also possible to find numerous areas of relevance that future research may pursue and

investigate. A theme that future researcher could also follow is the study of the advantages of using diverse types of programming languages, in order to teach the participants. A research of this nature may uncover whether one computer language in particular is the most appropriate for use with CSCL tools. Finally, an analysis of the wider hypothesis of using CSCL as programming teaching tools is more effective than other non-computer-supported collaborative learning approaches would also be important and could help to both enlighten and enhanced future teaching.

ACKNOWLEDGEMENTS

This research received funding from the Portuguese Foundation for Science and Technology (FCT) under grant LARSyS - UIDB/50009/2020.

REFERENCES

- Anderson, L. W., Krathwohl, D. R., Airasian, P. W., Cruikshank, K. A., Mayer, R. E., Pintrich, P. R., ... & Wittrock, M. C., 2001. A taxonomy for learning, teaching, and assessing: A re-vision of Bloom's taxonomy of educational objectives, abridged edition. White Plains, NY: Longman.
- Angeli, C., Valanides, N., Polemitou, E., & Fraggoulidou, E., 2016. An interaction effect between young children's field dependence-independence and order of learning with glass-box and black-box simulations: Evidence for the malleability of cognitive style in computer-supported learning. *Computers in Human Behavior*, 61, 569-583.
- Bruce, C. S., & McMahon, C. A., 2002. Contemporary developments in teaching and learning introductory programming: Towards a research proposal. Faculty of Information Technology, QUT Teaching & Learning Report, (2/2002).
- Byrne, P., & Lyons, G., 2001, June. The effect of student attributes on success in programming. In *ACM SIGCSE Bulletin* (Vol. 33, No. 3, pp. 49-52). ACM.
- Collis, K. F., & Biggs, J. B., 1979. Classroom Examples of Cognitive Development Phenomena: the SOLO Taxonomy.
- Dehnadi, S., 2006, September. Testing programming aptitude. In *Proceedings of the 18th Annual Workshop of the Psychology of Programming Interest Group*, pp. 22-37).
- Dybå, T., & Dingsøyr, T., 2008. Empirical studies of agile software development: A systematic review. *Information and software technology*, 50(9-10), 833-859.
- Estácio, B., Oliveira, R., Marczak, S., Kalinowski, M., Garcia, A., Prikładnicki, R., & Lucena, C., 2015, September. Evaluating collaborative practices in acquiring programming skills: Findings of a controlled experiment. In *Software Engineering (SBES), 2015 29th Brazilian Symposium on* (pp. 150-159). IEEE.
- Gilb, T., 2005. *Competitive engineering: a handbook for systems engineering, requirements engineering, and software engineering using PLanguage*. Elsevier.
- Gray, W. D., Goldberg, N. C., & Byrnes, S. A., 1993. Novices and programming: Merely a difficult subject (why?) or a means to mastering metacognitive skills? Review of the book *Studying the Novice Programmer*. *Journal of Educational Research on Computers*, 9(1), 131-140.
- Kitchenham, B., Pretorius, R., Budgen, D., Brereton, O. P., Turner, M., Niazi, M., & Linkman, S., 2010. Systematic literature reviews in software engineering—a tertiary study. *Information and software technology*, 52(8), 792-805.
- Jenkins, T., 2002, August. 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, No. 2002, pp. 53-58).
- Kolb, D., 1985. *Learning Style Inventory*. Hay/McBer, Boston, Ma.
- Lister, R., 2000, December. On blooming first year programming, and its blooming assessment. In *Proceedings of the Australasian conference on Computing education* (pp. 158-162). ACM.
- Lister, R., Simon, B., Thompson, E., Whalley, J. L., & Prasad, C., 2006. Not seeing the forest for the trees: novice programmers and the SOLO taxonomy. *ACM SIGCSE Bulletin*, 38(3), 118-122.
- Lahtinen, E., Ala-Mutka, K., & Järvinen, H. M., 2005. A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin*, 37(3), 14-18.
- Nosek, J. T., 1998. The case for collaborative programming. *Communications of the ACM*, 41(3), 105-108.
- Reiser, B.J., Anderson, J.R. and Farrell, R.G., 1985, August. Dynamic Student Modelling in an Intelligent Tutor for LISP Programming. *IJCAI* (V. 85, p. 8-14).

APPENDIX

- [DING17] Ding, Qing, and Sitan Cao. "RECT: A cloud-based learning tool for graduate software engineering practice courses with remote tutor support." *IEEE Access* 5 (2017): 2262-2271.
- [DONG17] Dong, Zhijiang, Cen Li, and Roland H. Untch. "Build peer support network for CS2 students." In *Proc. of the 49th Annual Southeast Regional Conference*, pp. 42-47. ACM, 2011.
- [GOLDMAN11] Goldman, Max, Greg Little, and Robert C. Miller. "Real-time collaborative coding in a web IDE." In *Proc. of the 24th annual ACM symp. on User interface software and technology*, pp. 155-164. ACM, 2011.
- [GUO15] Guo, Philip J., Jeffery White, and Renan Zanelatto. "Codechella: Multi-user program

visualizations for real-time tutoring and collaborative learning." In *Visual Languages and Human-Centric Computing (VL/HCC)*, 2015 IEEE Symp. on, pp. 79-87. IEEE, 2015.

- [HARSLEY17] Harsley, R., Fossati, D., Di Eugenio, B., & Green, N. (2017, March). Interactions of individual and pair programmers with an intelligent tutoring system for computer science. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (pp. 285-290).
- [JENKINS12] Jenkins, Jam, Evelyn Brannock, Thomas Cooper, Sonal Dekhane, Mark Hall, and Michael Nguyen. "Perspectives on active learning and collaboration: JavaWIDE in the classroom." In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, pp. 185-190. ACM, 2012.
- [JOVANOVIĆ15] Jovanovic, Dusan, and Slobodan Jovanovic. "An adaptive e - learning system for Java programming course, based on Dokeos LE." *Computer Applications in Eng. Education* 23, no. 3 (2015): 337-343.
- [LEUNG07] Leung, Chi-Hong, and Yuen-Yan Chan. "Knowledge management system for electronic learning of IT skills." In *Proceedings of the 8th ACM SIGITE conference on Info. technology education*, pp. 53-58. ACM, 2007.
- [NOGUEZ07] Noguez, Julieta, Gilberto Huesca, and L. Enrique Sucar. "Shared learning experiences in a contest environment within a mobile robotics virtual laboratory." In *Frontiers In Education Conference-Global Engineering: Knowledge Without Borders, Opportunities Without Passports, 2007. FIE'07. 37th Annual*, pp. F3G-15. IEEE, 2007.
- [STARBIRD11] Starbird, Kate, and Leysia Palen. "More than the usual suspects: the physical self and other resources for learning to program using a 3D avatar environment." In *Proc. of the 2011 iConference*, pp. 614-621. ACM, 2011.
- [TSOMPANOUDI13] Tsompanoudi, Despina, Maya Satratzemi, and Stelios Xinogalos. "Exploring the effects of collaboration scripts embedded in a distributed pair programming system." In *Proc. of the 18th ACM conference on Innovation and technology in computer science education*, pp. 225-230. ACM, 2013.
- [WANG09] Wang, Shu-Ling, Gwo-Haur Hwang, Ju-Chun Chu, and Pei-Shan Tsai. "The role of collective efficacy and collaborative learning behavior in learning computer science through CSCL." *ACM SIGCSE Bulletin* 41, no. 3 (2009): 352-352.