

Innovative Approaches in Teaching Programming: A Systematic Literature Review

Simone C. Santos^a, Patricia Azevedo Tedesco^b, Matheus Borba and Matheus Brito
*Centro de informática, UFPE, Rua Jornalista Anibal Fernandes, s/n, Cidade Universitária (Campus Recife),
CEP: 50.740-560, Recife, PE, Brazil*

Keywords: Educational Innovation, Teaching Computer Programming, Systematic Literature Review.

Abstract: One of the main challenges of computing education is the teaching of computer programming. Technical skills related to algorithm logic, programming language syntax, and computational platforms are required to program. In addition, several non-technical skills are required, enabling the student to understand and to interpret real problems, to work in groups and to strive for effective and efficient solutions. To meet these challenges, innovative teaching methodologies have been applied in teaching programming, building learning environments that are more conducive to the development of these skills. In order to understand how these methodologies are being used, this work presents the result of a systematic literature review, motivated by the following research question: "What are the innovative teaching and learning approaches to programming, how are they applied and what are the main results of their application? In this study, we considered three digital libraries and found 24 primary studies, following the Kitchenham methodology. These studies were categorized into 6 groups and highlighted challenges related to the problems addressed, teaching environment, content, human capital involved and assessment process. The studies also showed evidence of success cases, as well as open paths for new research.


1 INTRODUCTION


The evolution of the area of Computing over the last years is well-known. Computing is everywhere, whether to make life easier for people or to make businesses and organizations more efficient and competitive. It is responsible for automating processes, maximizing productivity, expanding communication, enabling better products and services, allowing the world to be more productive and agile. In this light, the trend is for this area to evolve further, requiring qualified professionals to perform better. This entails that Computing education follows the evolution of the area, aligning the academy with the demands of the labor market.

When it comes to computer education, one of the main challenges is teaching programming. This is compounded by the need to teach programming languages and their syntax, that allow instructional communication with computing machines.

Programming is not an easy subject. According to Lahtinen, Ala-Mutka, and Järvinen (2005), learning programming requires correctly understanding abstract concepts, mastering the syntax of languages that often differ from natural language, and logical reasoning to transform instructions into required computer actions. Many students experience learning problems due to the abstract nature of the subject or lack of enough resources to ensure personal support from teachers. With regards to group work, there are also difficulties, many of them concerning the size of the class versus the size of the teams, the heterogeneity of the group members, making project practices where everyone learns difficult. Consequently, high drop-out rates are often recorded in programming courses (Lahtinen, Ala-Mutka, and Järvinen, 2005).

To overcome these challenges, the authors of this article believe that one of the fundamental components in improving the quality of teaching programming is the learning methodology employed.

^a  <https://orcid.org/0000-0002-7903-9981>

^b  <https://orcid.org/0000-0001-9450-9219>

The use of new teaching and learning strategies can be a way of minimizing the challenges found more often, attracting more students to the computing area.

In Costelloe (2016), the author defines a number of approaches for teaching programming, namely: 1) Lectures & Labs; 2) Software Visualization; 3) Robots; 4) Problem-Based Learning; 5) Cognitive Apprenticeship; 6) Miscellaneous. Using these categories as a key reference, this study defined the central research question: "CQ - *What are the innovative teaching and learning approaches to Programming, how are they applied and what are the main results of their application?*".

In order to investigate possible answers to this question, the Systematic Literature Review (SLR) method proposed by Kitchenham in (Kitchenham, 2004) was used.

This paper is divided into five sections. After this brief introduction, Section 2 presents the main theoretical references used to structure the research and its analysis. Section 3 describes the application of the SLR introduced in (Kitchenham, 2004). Section 4 presents and discusses the results found, and finally Section 5 comments the conclusions and limitations of this study as well as future works.

2 MAIN THEORETICAL REFERENCES

Two main references are used to structure this study: 1) different approaches to teaching programming, according to (Costelloe, 2016) and; 2) a PBL (Problem Based Learning) methodology used to organize the report of results found, according to (Santos, Furtado, and Lins, 2014). We have chosen to use the Costelloe reference only because it is an in-depth and detailed study of the main categories of innovative approaches to teaching programming, and therefore a source of references in itself.

2.1 The State of the Art of Teaching Programming

The benefits brought about by recent advances in research and experience in teaching multilevel programming show different gains from each approach used, as well as various uses and impacts of such approaches. Thus, research has been done to improve teaching and learning programming. As a result, Costelloe (2016) presents several approaches developed, stemming from the technology-driven that involves the use of visual software, design and

robotics tools to education driven by paradigms such as PBL and cognitive learning.

According to Costelloe (2016), these approaches are categorized into six groups: 1) *Lectures and labs*; 2) *Software visualization*; 3) *Use of robots*; 4) *Problem based learning*; 5) *Cognitive apprenticeship* and; 6) *Miscellaneous approaches*.

The traditional approach of *Lectures and labs* adopts a behavioral theory of learning. The main applied behavioral theory is operant conditioning, where the student learns as a result of reinforcement, whether positive or negative. The students can be motivated by the teacher's humor, drama, enthusiasm and knowledge and understanding they acquire.

Software visualization is the practice of mapping abstract ideas represented in code by visual representations that make the operation of the system easier for the observer (Ayrapetov and Graham, 2002). In other words, software visualization is used to assist the programmer/user of a program to understand the artifact being observed. In Costelloe (2016), the author categorizes software visualization as follows: *Programs View*, focusing on the graphical representation of a running program and its data; *Algorithm Animation*, testing instructional use and showing the fundamental operations of an algorithm; *Visual programming*, with visual components to build a program; *Demonstration programming*, using Artificial Intelligence programming languages; and *Computational Visualization*, viewing statistics, for example, access points in the code in terms of counting errors, viewing statistics, for example, access points in the code in terms of counting errors.

Research shows that active learning, that is, learning promoted by interaction with the environment, as opposed to lectures, is more effective in developing the student's ability to acquire knowledge (Linder et al., 2001). Linder argues that active learning can be facilitated by the use of *mobile robots* in a collaborative environment. The main benefits of the use of Robots are (Costelloe, 2016): it promotes active learning, involving the student and promoting enthusiasm and fostering the learning processes; it promotes collaboration and the robot becomes a participant in this collaboration; it provides experience with real machines; it promotes creativity; students can generate hypotheses and test them by getting feedback immediately; it fosters good design and planning; it promotes leadership from practice, which promotes autonomous learning.

The *Problem-Based Learning* approach (PBL) concerns a constructivist view of learning, in which students adjust to existing belief constructs. According to Costelloe (2016), the main benefits of

using PBL are: to promote lifelong learning techniques, forcing the student to reflect about their learning process and to re-evaluate, through the maintenance of journals and learning portfolios; to promote understanding through collaborative work; to foster learning that driven by problems, rather than contents, and to be an approach that reflects real-life problems, preparing students for the job. In such an approach, problems can range from structured to poorly structured to meet the needs of beginners and advanced students. PBL promotes creativity in obtaining solutions to problems; promotes independent learning and compels students to take on responsibility for their work; promotes positive feelings about the course; and students learn other skills not specific to course, oral, writing of reports, demonstration, and critical thinking.

Cognitive apprenticeship is a model of collaborative teaching where the emphasis is on supporting the construction of knowledge (Enkenberg, 2001). According to Enkenberg, cognitive apprenticeship applies several strategies of teaching and learning: Modeling, that is, the demonstration of thought processing; Explanation, because activities happen as they do; Scaffolding, that entails supporting students to handle the task at hand, and gradual withdrawal of teacher from the process; Reflection, Self-assessment and Self-analysis; Articulation, which entails reflection results placed in verbal form; Exploration, where students are encouraged to form hypotheses, test them and find new ideas. Similar to PBL, the basic principles of cognitive apprenticeship relate to constructivism, to the student constructing their own knowledge, aided by a specialist initially and gradually becoming an independent learner, the aspect of collaboration and the development of metacognitive skills to reflect about their work.

Finally, these approaches can be combined in order to help them adapt to specific learning contexts and obtain diverse benefits from each one. This combination is pointed out as a "*miscellaneous approach*".

2.2 Methodological Elements

This study has used the PBL methodology elements as a theoretical background for analysing the results of the SLR. In Santos, Furtado, and Lins (2014), the authors propose a methodology for the implementation of PBL in Computing based on 5 manageable elements: 1) *Problem*, reflecting realism and complexity similar to real contexts; 2) *Environment*, related to the definition of an authentic

learning environment that reflects the actual context of the professional market; 3) *Human Capital*, with evidence to the roles and responsibilities of the pedagogical team in the planning; 4) *Content*, as an essential part to support the theoretical basis of the problem solving process; and 5) *Processes*, for the adequacy of learning objectives and assessment processes inherent to the learning format in PBL. We will use these elements as reference in the analysis described in Section IV, particularly in the results of question RQ4.

3 RESEARCH METHOD

According to Kitchenham (2004), Systematic Literature Reviews (SLR) is a method designed to identify, evaluate and interpret all available research relevant to a particular research question, area, topic, or a phenomenon of interest. We guided our investigation on SLR by the procedures also defined by Kitchenham. The common reasons for undertaking a systematic literature review are to summarise the existing evidence concerning a topic of interest and to identify any gaps in current research in order to suggest areas for further investigation.

The individual studies that contribute to a systematic review are called primary studies; the systematic review by itself is a form of a secondary study. The systematic review conducted in this study was divided into three phases, based on Kitchenham's guide, namely: Planning the review; Conducting the review and Reporting the review.

The planning and conducting phases have several smaller phases, while reporting is a single phase. The Planning stage can be divided into two main parts. These can be further detailed as: the *Objective of the systematic review*; Development of a *review protocol*, that includes the definition of the research questions (question types; question structure).

The Conducting phase can be divided into five main parts. Within each one we have more specific phases, which are: *Identification of Research*; *Study Selection*; *Study Quality Assessment*; *Data Extraction*; *Data Synthesis*.

This systematic review should answer the following central question: "CQ - *What are the innovative teaching and learning approaches to Programming, how are they applied and what are the main results of this application?*". More specifically, this review focuses on the following issues:

- **RQ1:** What is the teaching context (educational stage)?

- **RQ2:** What are the innovative approaches of the studies?
- **RQ3:** What are the main results found from these approaches?
- **RQ4:** What are the main challenges or difficulties found in general way?

The initial research studies were conducted using digital libraries: IEEEExplore Digital Library, ACM Digital Library, and Science Direct. The selection of these sources was based on the credibility and relevance of articles indexed in the area of Computer Science. After selecting our electronic databases, the search string was created and refined until it was established and could be used on all bases. By combining keywords and synonyms the search string was constructed, as shown:

("teaching" OR "learning") AND ("programming" OR "programming languages") AND ("innovative" OR "innovation") AND ("case study" OR "case studies" OR "lessons learned" OR "experience report")

The research strategy carried out considered papers published between the years of 2012 and the first half of 2018 and resulted in a total of 606 papers: ACM - 65 papers; IEEE - 2 papers; Science Direct - 535 papers;

We established a filter with exclusion criteria. If the paper fit into at least one, it would be eliminated. The following exclusion criteria were used:

- Exclude artifacts according to the analysis of titles and abstracts;
- Non-English articles;
- Articles with paid content;
- Duplicate, repeated articles;
- Articles not available for download or viewing.

With regards to the last exclusion criterium, we excluded articles not available for download or viewing at our institution. In the digital libraries considered, those often mean articles published in conference proceedings in recent years. After the application of this filter, 35 primary studies remained.

Then quality criteria were established; artifacts that did not refer to the study area or did not present good content completeness or clarity, or that did not answer the questions elaborated would be eliminated. The following are the quality criteria used:

- Relevance to the study area/central theme.
- Completeness and clarity of contents.
- If the study answered the questions.

3.1 Selecting Primary Studies

The initial analysis was based on the introductions and conclusions of the papers. After the application of this second filter, 24 papers were obtained. Using the same quality criteria, we have analyzed the artifacts by reading them in full: as a result, 20 papers remained. These papers were the selected works for extracting the answers and consolidating the results. The distribution between the databases is found below:

- ACM - 14 papers;
- IEEE - 0 papers;
- Science Direct: 6 papers.

After the automatic search, a manual search of papers was performed at conferences that were held in the second half of 2018. The papers selected went through all the pre-established steps and we had 4 more papers, all of them were presented in the FIE 2018 (<http://www.fie.org>). The total number of papers selected for analysis is shown in Table 1.

Table 1: Primary studies.

ID	Title, Author
PS1	Perspectives on active learning and collaboration: JavaWIDE in the classroom (Jenkins et al., 2012)
PS2	CS1001.py: a topic-based introduction to computer science (Chor, et al., 2012)
PS3	Course development through student-faculty collaboration: a case study (Ustek et al, 2014)
PS4	Smartphones, Studio-Based Learning, and Scaffolding: Helping Novices Learn to Program (Reardon and Tangney, 2014)
PS5	Teaching Software Engineering with LEGO Serious Play (Kurkovsky, 2015)
PS6	Teaching Java Programming on Smartphone-pedagogy and Innovation; Proposal of its Ontology Oriented Implementation (John and Rani, 2015)
PS7	Using Project-Based-Learning in a mobile application development course—An experience report (Francese et al., 2015)
PS8	Learning Basic Programming Concepts by Creating Games with Scratch Programming Environment (Ouahbi et al., 2015)
PS9	Combining mastery learning with project-based learning in a first programming course: an experience report (Jazayeri, 2015)
PS10	Building Casual Game SDKS for Teaching CS1/2: A Case Study (Sung et al., 2016)
PS11	Using Interactive Exercise in Mobile Devices to Support Evidence-based Teaching and Learning (Fuad et al., 2016)

Table 1: Primary studies (cont.).

ID	Title, Author
PS12	Applying Validated Pedagogy to MOOCs: An Introductory Programming Course with Media Computation (Falkner et al., 2016)
PS13	Teaching DevOps and Cloud Computing using a Cognitive Apprenticeship and Story-Telling Approach, Christensen, 2016.
PS14	Visual programming languages integrated across the curriculum in elementary school: A two-year case study using “Scratch” in five schools, Sáez-López et al., 2016.
PS15	Teaching real-time programming using mobile robots*, Rodríguez et al., 2016.
PS16	Computing Curriculum in Middle Schools: An Experience Report, Sabbagh et al., 2017.
PS17	Computing for Medicine: An Experience Report (Campbell et al., 2017)
PS18	Teaching concurrent and parallel programming by patterns: An interactive ICT approach (Capel et al., 2017)
PS19	K-12 Teachers Experiences with Computing: A Case Study (Cooper et al., 2017)
PS20	Practical Robotics in Computer Science Using the LEGO NXT: An Experience Report (Estrada, 2017)
PS21	Applying PBL in Teaching Programming: an Experience Report (Santos et al., 2018)
PS22	Improving Student’s Learning and Cooperation Skills Using Coding Dojos (In the Wild!) (Matheus et al., 2018)
PS23	Student Experiences with Collaborative Problem-Based Learning (CPBL) in a Second-Year Undergraduate Engineering Course (Fang, 2018)
PS24	Inclusive Model for the Development and Evaluation of Accessible Learning Objects for graduation in Computing: A Case Study (Mourão and Netto, 2018)

Figure 1 shows the evolution of the studies over the last seven years (2012-2018). The curve of the graph of Figure 1 shows a growing trend in recent years, considering that the automatic search of RSL did not include the year 2018 in full.

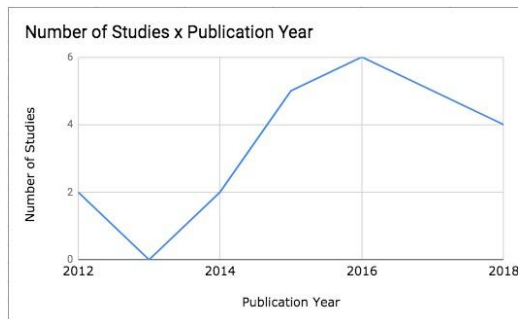


Figure 1: Temporal distribution of primary studies.

3.2 Limitations and Threats to Validity

This study has some limitations, which exist in any qualitative research. Considering the literature review, qualitative findings are highly based on context, and case-dependent. To avoid bias, well-established research methods with the support of data extraction artifacts based on a spreadsheet and tags definition were adopted, besides the direct participation of two specialists with vast experience in teaching computing together two undergraduate students in computer science.

4 RESULTS

The next subsections comment on the research questions.

4.1 Teaching Context

Out of the 24 studies, the most found context was undergraduate teaching (16 out of the 24 studies). Two studies talk about multiprogram teaching, which evidences the need for programming skills; three talk about middle to high school programming and two studies talk about teaching programming for other courses, such as medicine, administration and economy. The following quotes evidence some of these results:

"This study relates a multi-program (high school summer enrichment courses, and at two- and four-year colleges), teaching Java Programming; in GA, USA". **PS1**

"This study relates a high school program that considers teaching; Basic programming concepts; the study took place at the Abdellah University, Morocco." **PS8**

"This study reports a computing course in the context of a Medical Doctor Program at the University of Toronto." **PS17**

Figure 2 shows a summary of the educational levels considered in the studies and their respective concentrations.

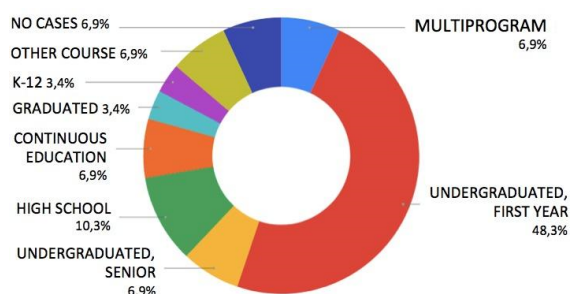


Figure 2: Educational levels considered in the studies.

We can see that the greatest concentration of studies is in adult education, particularly in the undergraduate stage.

4.2 Proposal of Studies

Considering the categories proposed in Costelloe (2016), the primary studies have shown evidence of the adoption of four categories: cognitive apprenticeship, PBL, robots, and approaches that involved a combination of these proposals, defined as “miscellaneous”. Figure 3 shows an overview of this adoption, with a greater number of evidences to the cognitive apprenticeship and PBL categories.

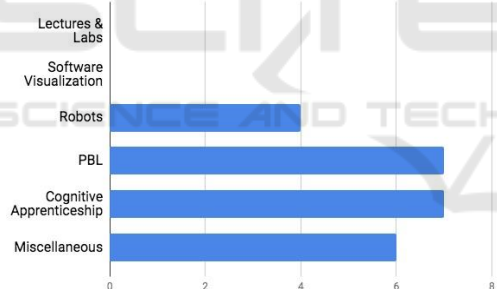


Figure 3: Proposal categories identified in the studies.

Table 2 summarizes the proposal of each category, identifying its respective studies.

With regards to languages and approaches, most studies use Java or Python, or, alternatively a visual programming approach. Some deal with game-based learning and some deal with project-based learning. Most studies deal with collaborative programming learning. Out of the 24 studies considered, three used app development as the main proposal; three used robot programming as their main approach, particularly with the Lego NXT robot kit. With respect to individual and collaborative learning, we list below some evidences of these results:

Table 2: Primary studies per categories.

Category	Proposal	Studies
Lectures& Labs	No evidence.	
Software Visualization	No evidence.	
Robots	Use of the LEGO technology (serious play, Mindstorms, NXT).	PS5, PS9, PS15, PS20
PBL	Project-based practices, in order to develop games, mobile apps, systems prototypes, and so on.	PS6, PS7, PS8, PS14, PS21, PS23, PS24
Cognitive Apprenticeship	Collaborative learning, technology support (such as games, 3D virtual environment), and scaffolding used to understand and discuss programming concepts.	PS1, PS2, PS10, PS12, PS13, PS16, PS18
Miscellaneous	All of them combining PBL and Cognitive Apprenticeship, and two of them also including robots (PS3 and PS9).	PS3, PS4, PS9, PS11, PS17, PS22

"How smartphones, studio-based learning, and extensive scaffolding were used in combination in the teaching of a freshman Introduction to Programming course, beginning with the visual programming environment Scratch and culminating with Java development for Android smartphones." **PS4**
"This paper discusses the use of the LEGO MINDSTORMS NXT mobile robots for teaching real-time programming to bachelor's students. The stable real-time control of a segway-like robot with a PID controller is used as case study to demonstrate the teaching methodology. Ada was used as real-time programming language." **PS15**

4.3 Main Results

This study used the elements of the PBL methodology as a theoretical basis for the analysis of the results found from the use of innovative approaches in programming teaching. Among the results found, we highlight the 9 categories as shown in Table 3.

Table 3: Main results from innovative approaches.

Main Results	Description	Primary Studies
Technical Skills	Syntax, semantic, core concepts, computational thinking, specific technologies	PS3, PS6, PS9, PS11, PS14, PS15, PS16, PS18, PS20
Soft Skills	Teamwork, oral communication, creativity, critical thinking, problem solving, management skills (time, product quality, requirements), self-directed	PS3, PS5, PS7, PS8, PS16, PS21, PS24
Collaboration /Cooperation	Collaborative learning, Peer review	PS3, PS4, PS6, PS23
Student satisfaction	Motivation, engagement, effective learning, better student performance, student confidence, improved retention	PS1, PS3, PS4, PS5, PS7, PS8, PS9, PS10, PS11, PS12, PS13, PS14, PS15, PS16, PS17, PS18, PS20, PS21, PS23, PS24
Real experiences	Real practices, create application, thoughts of entrepreneurship, sharing experience with software professionals	PS6, PS14, PS20, PS22, PS23, PS24
Content sharing	Software code, artifacts, apps, exercises	PS1, PS10, PS11, PS13
Assessment	Monitoring of the student progress, continuous feedback	PS1, PS7, PS11, PS21
Ample curriculum	Curriculum with ample vision of the CS area	PS2
Teaching training	Professional development to in-service K-12 teachers	PS19

The most discussed results highlighted the motivation and engagement of students through the use of innovative teaching approaches and strategies and intense collaborative work, always with great enthusiasm and recommendation of continuity of the respective proposals. The following evidences illustrate some of these results:

"The resulting course meets goals and objectives, provides wonderful motivation, and highlights creativity and intellectual challenge within computer science as well as syntax, semantics, and core technical skills." **PS3**

"In this pedagogical design, students interact and create their own content related to curricular areas with several advantages, such as motivation, fun, commitment, and enthusiasm, showing improvements related to computational thinking and computational practices." **PS14**

"This approach has been very efficient and our application and had a significant engagement of the participants in the course, considering the results obtained and the satisfaction of students in learning programming." **PS21**

Other results also highlighted student performance, pointing to considerable improvements in the development of both technical skills related to the knowledge of key elements in programming (syntax, semantics, basic and advanced concepts, platforms) and non-technical (professional skills) related to teamwork, communication, innovation, and managerial skills:

"That LSP has a positive impact on student learning, while also improving student engagement with the course material. Formal data along with written and informal feedback from students suggests that LSP helped improve soft skills, such as teamwork and oral communication." **P5**

"The student performance data shows its effectiveness in increasing student understanding of difficult concepts and the overall perception of using the software was very positive." **P11**

"Results show that students who studied computing through Alice ME showed an improvement in their critical thinking and problem-solving skills." **P16**

A special highlight was the construction of a practice-based learning environment, promoting more effective learning from real problems and solutions, as emphasized in following study:

"Outcomes from our experience can be considered positive: all the students delivered project artifacts on time, with a good level of quality and completeness with respect to the established requirements (as gathered values for software metrics also suggest). All the students were enthusiastic in developing apps for smart devices;(…)" **P7**

In addition to these results, some studies have highlighted the sharing of content produced by students, such as software code, documents, applications, and other artifacts, which are important in the learning process. It is also important to remark the importance of monitoring students' progress with continuous feedback. Two studies highlighted specific results, one aimed at proposing a broader CS curriculum, rather than an emphasis on programming alone; and another, focused on the training of K-12 teachers, with the aim of forming programming skills in students before higher education.

Finally, despite considering the challenges encountered in general, it was possible to identify how common they are in the categories most found in Table 2.

4.4 Main Challenges

Considering the positive results of innovations in teaching programming, not all studies have explicitly commented on the open challenges. Some studies highlighted improvements in their learning environments, problems related to teaching methodology planning, and others commented on the culture change related to collaborative work. In order to facilitate the understanding of the mentioned challenges, we grouped them according to the elements of the methodology defined in (Santos, Furtado, and Lins, 2014), commented briefly in Section 2.2. The main challenges are shown in Table 4.

Table 4: Main challenges in adopting innovative approaches.

Main Challenges	Description	Primary Studies
Problem	Timing considerations, Short time for practices, Task planning, Emphasize programming process and realistic context, Presence of computing in medicine	PS5, PS13, PS15, PS17, PS22, PS23
Environment	Configuration and Installation of programming environments, Maintenance/Evolution of specialized labs, Cost of technologies, Scaling up the course,	PS1, PS3, PS4, PS18, PS20
Content	Lack time to learning specific topics deeply, Didactic materials elaboration/ preparation, Complete pattern libraries using Java, C ++ and C # languages, Need to specifically scaffold the integration of concepts	PS2, PS3, PS12, PS15, PS21
Human Capital	Better to strong students than to weak students, Adopting faculty's experience and expertise in application domains, culture of sharing and reflective practices, Competition between students impacting participation, Team building, Some students prefer to study alone	PS7, PS9, PS10, PS16, PS21, PS22, PS23
Assessment Process	More rigorous evaluation tools, Accompaniment of the learning methodology	PS16, PS21

Regarding the Problem element, some studies have highlighted the need for special care with the definition of time to perform tasks, which depending on its complexity, can overwhelm the student, as in the quote below:

"Timing considerations are extremely important when planning an LSP activity that is supposed to fit

a single class period. All LSP case studies described here were designed for 75-minute blocks." **PS5**

Still on this element, the importance of problems arising from a real context, with the possibility of practical application, as shown in the quote below:

"As DevOps is highly skills-oriented and has strong requirements on bringing up complex deployment architectures fast, we have argued in favor of teaching methods that emphasize programming process and realistic context as well as using performant virtualization environments." **PS13**

Regarding the Environment, a special highlight is the need for configuration and installation of programming environments, maintenance and/or evolution of specialized labs, and the cost of technologies, such as robot development kits, as highlighted in *"we are looking at possibilities for scaling up the course. The current set of robot kits limits our enrollment to 66 students, adding NXT is slightly complicated by the fact that the NXT v2.0 has now been discontinued."* **PS20**

This evolution of environments is also related to two challenges highlighted: high costs and difficulty in scaling up the course.

Regarding the Content element, some studies have highlighted the need to develop better and more complete teaching materials, including the need for strategies that can promote the integration of contents in an easier way for students. Study (Falkner et al., 2016) highlights this need:

"These results support a hypothesis of the need to specifically scaffold the integration of concepts, beyond their mastery as individual concepts." **PS12**

Challenges related to the Human aspect are also commented on. In (Jazayeri, 2015) the study emphasizes that the approach favors students who are stronger over the weaker ones:

"Our combined approach helped the strong students more than the Finaweak students." **PS9**

Although some studies have shown concern about the formation of teams aiming at collaborative work (Santos et al., 2018), other studies point to characteristics that hinder collaboration, such as competitiveness, making it difficult to share artifacts, and preference for individualized study:

"students 1) are stuck on a problem due to not understanding it, 2) lack support from other students (in terms of efforts and expertise), 3) have not enough time to solve problems, and 4) prefer to study alone." **PS23**

From the point of view of teachers, the study performed by Sung, Nash, and Pace (2016) emphasizes the importance of adopting the

experience and knowledge of faculty in application domains.

Lastly, but by no means least, as far as the assessment process is concerned, although it is a critical subject in any discussion on education, few studies have highlighted it as being a challenge. One study highlighted the importance of a more rigorous student follow-up process (Sabbagh et al., 2017); and another study highlighted the need for management of learning methodology, always aiming for improvements (Santos et al., 2018).

5 CONCLUSIONS

The evolution of computing in all areas of daily life, over the last decades is very well documented. This entails that the teaching of computing also evolves, and is kept synchronized with the needs of the market. However, teaching and learning Programming, one of the core subjects in computing are not easy tasks (Lahtinen, Ala-Mutka, and Järvinen, 2005). The authors of this study argue that one key aspect to be considered is the teaching methodology used. In this light, this study has set out to investigate what are the innovative teaching and learning aspects found in the literature.

To this end, we have carried out a SLR that aimed at finding out what are the innovative teaching and learning methods found in the literature, identifying the educational context, the results and challenges of their application. The review was executed in three Computer Science Digital libraries.

Twenty-four primary studies were found in the literature. Out of those, eighteen reported investigations concerning undergraduate programming learning, in response to **RQ1**. As far as languages are concerned, most studies deal with Java or Python. There is also a trend to use apps, robots programming, and most experiences also are situated, using either Project-based learning or Problem-based Learning (**RQ2**). It is also reported that the intensive use of collaborative learning, together with the continuous monitoring of the students' progress fostered motivation and engagement and eagerness to take projects further (**RQ3**).

Some of the challenges (**RQ4**) found involve structuring and scaling up courses, which is related to better structuring the learning environments available; lack of time to learn some concepts more deeply; time demanded in planning the course and developing appropriate pedagogical materials. In this respect we have also found reports of difficulties to integrate teaching and learning resources that are

available on the web. There are also challenges related to human capital and teamwork. Challenges related to human capital encompass finding ways to cater for both the stronger and weaker students; it also shows the need to integrating the faculty's expertise in real world domains in the classroom, promoting reflective and collaborative practices. There is also a need for more evaluation tools and monitoring of the learning methodologies used.

Finally, the authors believe that the categories presented in this paper present a good starting point for researchers in the area. In the near future, the authors intend to focus on the developing and assessing new methodologies, and that better assessment of the application of teaching methodologies is needed.

Starting with studies that combine approaches such as PBL and cognitive apprenticeship, one of the research proposals is to define a teaching methodology to Python programming for high school students, a gap that was identified. Each year, our institution promotes this training based on the PBL approach, during student vacations. The aim is to motivate students with the area of computing, from the development of programming skills. To this end, we will be analyzing the studies focusing on how to plan and manage all elements of innovative approach (problem, environment, content, human capital, and evaluation process).

ACKNOWLEDGEMENTS

The results presented here were developed as part of joint research groups: N.E.X.T (*iNnovative Educational eXperience in Technology*) research group, with a focus on Computing Education Research (CER); "PET Informática", a tutorial education program created by Brazilian government with the aim of developing standards of quality and academic excellence, highlighting the articulation of teaching, research and extension; and i-team. All of these groups work at "Universidade Federal de Pernambuco", Brazil. Many thanks for all involved in this study.

REFERENCES

- Al Sabbagh, S., H. Gedawy, H. Alshikhabobakr, S. Razak, 2017. "Computing Curriculum in Middle Schools: An Experience Report".
- Ayrapetov, D., S. Graham, 2002. "Program Visualisation: Cognitive Issues and Technological Implementations".

- Campbell, J., M. Craig, M. Law, 2017. "Computing for Medicine: An Experience Report".
- Capel, M., A. Tomeu, A. Salguero, "Teaching concurrent and parallel programming by patterns: An interactive ICT approach", 2017.
- Christensen, H. B., 2016. "Teaching DevOps and Cloud Computing using a Cognitive Apprenticeship and Story-Telling Approach".
- Cooper, S., S. Rodger, K. Isbister, M. Schep, R. Stalvey, L. Perez, 2017. "K-12 Teachers Experiences with Computing: A Case Study".
- Costelloe, E., 2016. "Teaching Programming: The State of the Art. The Center for Research in IT in Education".
- Enkenberg, J. , 2001. "Instructional design and emerging teaching models in higher education".
- Estrada, Francisco J., 2017. "Practical Robotics in Computer Science Using the LEGO NXT: An Experience Report".
- Falkner, K., N. Falkner, C. Szabo, R. Vivian, 2016. "Applying Validated Pedagogy to MOOCs: An Introductory Programming Course with Media Computation".
- Fang, N., 2018. "Student Experiences with Collaborative Problem-Based Learning (CPBL) in a Second-Year Undergraduate Engineering Course".
- Francese, R., C. Gravino, M. Risi, G. Scanniello, G. Tortora., 2015. "Using Project-Based-Learning in a mobile application development course—An experience report".
- Fuad, M. Muztaba, D. Deb, J. Etim, C. Gloster, 2016. "Using Interactive Exercise in Mobile Devices to Support Evidence-based Teaching and Learning".
- Hod, B. Chor R. et al., 2012. "CS1001.py: a topic-based introduction to computer science".
- Jazayeri, Mehdi, 2015. "Combining mastery learning with project-based learning in a first programming course: an experience report".
- Jenkins, J., E. Brannock, T. Cooper, S. Dekhane, M. Hall, M. Nguyen, 2012. "Perspectives on active learning and collaboration: JavaWIDE in the classroom".
- John, Mr. Santhosh, Dr. Mary Shanthi Rani, 2015. "Teaching Java Programming on Smartphone-pedagogy and Innovation; Proposal of its Ontology Oriented Implementation".
- Kitcheham, B., 2004. "Procedures for Performing Systematic Reviews".
- Kurkovsky, S., 2015. "Teaching Software Engineering with LEGO Serious Play".
- Lahtinen, E., K. Ala-Mutka, H. Järvinen, 2005. "A Study of the Difficulties of Novice Programmers".
- Linder, S. Paul, B. Edward Nestricks, S. Mulders, C. Leah Lavelle, 2001. "Facilitating Active learning with inexpensive Mobile Robots".
- Manuel, J., Sáez-López, M. Román-González, E. Vázquez-Cano, 2016. "Visual programming languages integrated across the curriculum in elementary school: A two year case study using 'Scratch' in five schools".
- Mourão, Andreza Bastos, José Francisco Magalhães Netto, 2018. "Inclusive Model for the Development and Evaluation of Accessible Learning Objects for graduation in Computing: A Case Study".
- Oliveira, C. Matheus Campos de, E. Canedo, H. Faria, L. Henrique Vieira Amaral, R. Bonifacio, 2018. "Improving Student's Learning and Cooperation Skills Using Coding Dojos (In the Wild!)".
- Ouahbi, I., F. Kaddari, H. Darhmaoui, A. Elachqar, S. Lahmine, 2015. "Learning Basic Programming Concepts by Creating Games with Scratch Programming Environment".
- Reardon, S., Brendan Tangney, 2014. "Smartphones, Studio-Based Learning, and Scaffolding: Helping Novices Learn to Program".
- Rodríguez, C., J. Luis Guzmán, M. Berenguel, 2016. "Teaching real-time programming using mobile robots".
- Santos, S. C., E. Santana, L. Santana, P. Rossi, L. Cardoso, U. Fernandes, C. Carvalho, and P. Tôrres. 2018. "Applying PBL in Teaching Programming: an Experience Report", FIE, San Jose, USA.
- Santos, S. C., Furtado F., Lins W, 2014. "xPBL: a Methodology for Managing PBL when Teaching Computing", FIE, Madrid, Spain.
- Sung, K., R. Nash, J. Pace, 2016. "Building Casual Game SDKS for Teaching CS1/2: A Case Study".
- Ustek, D., E. Opavsky, H. Walker, D. Cowden, 2014. "Course development through student-faculty collaboration: a case study".