Computer-supported Collaborative Learning in Programming Education: A Systematic Literature Review

Leonardo Silva
Dep. of Informatics Engineering of
University of Coimbra
Coimbra, Portugal
leonardo.silva@garanhuns.ifpe.edu.br

António José Mendes
Dep. of Informatics Engineering of
University of Coimbra
Coimbra, Portugal
toze@dei.uc.pt

Anabela Gomes
Dep. of Informatics Engineering of
University of Coimbra
Coimbra, Portugal
anabela@isec.pt

***Abstract*—Social dimension plays a crucial role in the learning process. The use of technological resources to stimulate and mediate the interaction between students is known as Computer-supported collaborative learning (CSCL). Despite being widely used in programming education, the summarization of the academic literature about this topic is scarce. To create that body of knowledge, a systematic literature review was performed. The findings provide an understanding of how collaboration is explored in introductory programming, resources used to stimulate them and challenges in the process. Opportunities for future research are discussed, especially related to motivation, self-efficacy and engagement in CSCL, and the exploitation of learning analytics.**

***Index Terms*—computer-supported collaborative learning, programming learning, introductory programming.**

I. Introduction

Computer programming education is known to be difficult [1]. As a consequence, a high number of dropouts and unmotivated students are found [2]. Despite many research studies over the decades, problems persist.

Many solutions have been proposed, and the use of computers in education (Computer-based Learning Environment - CBLE) is one example, with a growing body of statistical evidence that supports its efficacy [3]. This review will focus on one special case of CBLE, the Computer-supported Collaborative Learning (CSCL) [4]. It is grounded in the Social Constructivism Theory from the Soviet psychologist Lev Vygotsky, on the assumption that learning is essentially a social activity. In that context, CSCL uses technology to mediate and stimulate interaction between students, and over the years researchers found evidence that confirms the effectiveness of this educational strategy in multiple domains [5].

However, despite being widely used and researched, there is a lack of systematic reviews summarizing the findings of CSCL in programming education. This absence may impact the adoption of this technology by teachers, as literature reviews and meta-analysis represents an important source of data to justify pedagogical decisions [6]. Also, findings from these studies provide insights for new research and present limitations of current literature.

This study presents the results of a systematic literature review performed on this subject. This paper is organized as follows. Section II presents a literature review; Section III describes the methodology adopted in this study; Section IV reports the results and analysis. And, finally, Section VI presents the conclusions of this study.

II. Literature Review

The decision to adopt a particular teaching strategy or instructional design must be based on rigorous empirical data, especially those coming from scientific experiments. This practice is known as evidence-based education and provides a solid foundation to mitigate problems in this context [6]. However, it depends on the availability of scientific data, e.g. systematic reviews and meta-analysis.

Although CSCL in programming education has been used for years, there is a scarcity of reviews on this topic. Related studies have not focused exclusively on this topic, and reported this area as part of a broader subject. Therefore they were unable to address the specifics of the topic or missed relevant papers.

For example, a review of technology and non-technology collaboration was performed in [7]. Although they included some papers related to CSCL for introductory programming, twenty-two studies analyzed in our work, were not reviewed. Differences in the review protocol affected it, keywords, database selection, research questions and the date range of studies. Also, we did explore how collaborative work as measured.

A systematic mapping of CSCL in the domain of software engineering education (SEE) was performed in [8], and studies related to programming education were included as a subcategory. Besides the differences in the types of studies (mapping vs literature review), which guided different research questions, the year range of selected studies (2003 to 2013), differs from ours (2015-2019).

The aforementioned differences between reviews and their scopes left specific questions on CSCL for programming learning without answers.

III. Method

This systematic review followed Kitchenham's methodology as it is extensively used in the computer science research [9]. The following steps were conducted: a) development of a research protocol; b) execution of the review; c) data synthesis and analysis, and d) review reporting.

A. Research protocol

1) Research question: This review aims to understand how CSCL was exploited in introductory programming education. Five research questions were constructed: a) what collaborative resources were used?; b) which resources are most effective? c) what has been measured in those experiments? d) how collaboration is structured? e) how collaborative work was measured?

2) Keywords and database selection: The keywords: cscl ["learning programming" OR "programming learning" OR "teaching programming" OR "programming teaching" OR "novice programmer"] were used to search on Google Scholar Database. Papers were filtered by date (2015 and above) and only peer-reviewed conferences/journals and Ms.C dissertations/Ph.D. theses were considered.

3) Inclusion criteria: Additional inclusion criteria were defined: i) pedagogical context must be related to the introductory programming concepts, as described in the section SDF/Fundamental Programming Concepts of the ACM Computer Science Curricula, they include: variables, input/output, conditionals and interactive structures, functions and recursion [10]. Studies in multiple academic-levels were included, from K-level (1-12) to undergraduate; ii) students must have collaborated through technological resources (CSCL) and iii) must be a quasi-experimental, experimental or observational study.

B. Quality analysis

To assess the quality of experimental studies included in this review, they were checked against the experimental design recommendations proposed in [11]. The considered criteria were: Q1) research objectives definition; Q2) sample description; Q3) description of the experimental context (in this study it is related to programming education); Q4) description of the experimental design; Q5) use of randomized controlled trial; Q6) use of pre/post testing; Q7) use of control/experimental group.

For each study, the adequacy for each criterion was classified as yes (Y), no (N) and partially (P). For example, studies that described only the age of participants, not mentioning gender, previous knowledge in programming and other relevant information, received a "partially" classification in Q2.

C. Review execution

The review process was performed in two phases. First, title, abstract and keywords were analyzed to find studies that met inclusion criteria. Secondly, a full paper read was performed on the selected papers, excluding duplicates and checking against the inclusion criteria once again.

Fig. 1. Systematic literature review process.

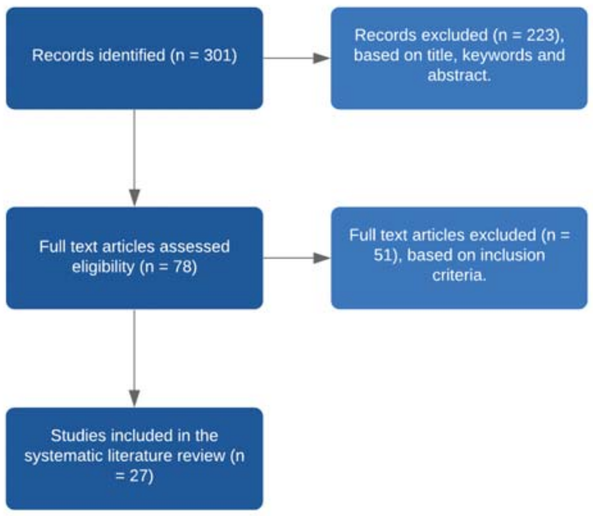
Records identified (n = 301)
Records excluded (n = 223), based on title, keywords and abstract.
Full text articles assessed eligibility (n = 78)
Full text articles excluded (n = 51), based on inclusion criteria.
Studies included in the systematic literature review (n = 27)

TABLE I
Excluded papers.

Papers	Reason
[19]; [20]; [21]; [22]; [23]; [24]; [25]; [26]; [27]; [28]; [29]; [30]; [31]	Not a CSCL environment.
[32]; [33]; [34]; [35]; [36]; [37]; [38]; [39]; [40]; [41]; [42]; [43]; [44]; [45]; [46]; [47]; [48]; [49]	Did not performed an experiment or observational study.
[50]; [51]; [52]; [53]; [54]; [55]; [56]; [57]; [58]; [59]; [60]	Not in the context of introductory programming.
[61]; [62]	Not accessible.
[33]; [63]; [64]; [65]; [66]	Duplicated.
[67];	Not a peer-reviewed conference/journal.

IV. Results and analysis

Three hundred and one studies were analyzed in the first phase, seventy-eight studies were selected for full-text reading, with fifty-one studies discarded because they did not meet one or more of the inclusion criteria (Figure 1). Table I presents the papers that were discarded and the justification. Twenty-seven studies were analyzed in this systematic review and results are summarized in Table II.

Overall, CSCL was used in a variety of educational levels. Robotics and tangible gadgets were mainly used by children, and software was mostly adopted in undergraduate settings.

Social interaction present in these environments encourages learners to use a higher-level of cognitive strategies [12]; [13] and develop logical-thinking [14]. Motivation [15] and engagement [16] were also benefited. However, fewer studies have investigated these variables.

Interventions happened in laboratory or in conjunction with classroom lectures [15]; [17]; [18]. Also, there were experiments where students were not physically co-located.

TABLE II
SELECTED PAPERS FOR REVIEW.

Paper	Description	Results
[17]	Wiki resources: shared edition, modification history and forum, were used to simulate a collaborative programming environment.	Experimental group had better results, however control group had less stress during activity.
[68]	An Eclipse plugin with shared editor, pair-programming roles, chatting, among others features.	Experimental group had better pass rate than control group.
[15]	A group of students worked on programming exercises using a voting system. A mobile application was used to propose and vote for a solution provided by their peers.	Learning and motivation gains.
[69]	Moodle resources (forum and Wiki) were used to support collaborative discussion; A Moodle Plugin called VLP allowed collaborative programming exercises	Wiki and forum did not show learning benefits, but the suggested platform increased collaboration.
[18]	Mobile application with augmented reality was used to stimulate collaboration in programming exercise.	Learning gains and collaboration were promoted.
[14]	CSCL system supported by discussion boards and group discussion scaffolding.	Experimental group presented better results in the logical post-test.
[12]	Wiki resources from Moodle to support collaborative programming exercises, similar to [17].	Experimental group had better learning and self-efficacy results, than the control group.
[70]	The system presents programming exercises with errors and a group of students should detects and corrects them.	No differences were found between experimental and control group.
[13]	Students worked in groups to solve programming problems with Scratch.	Gains in high-order thinking when compared with the control group.
[16]	Robotic and Scratch were used to promote children's collaborative programming tasks.	Students were engaged in programming tasks.
[71]	Use of Android App Inventor in pairs (experimental) and solo (control group).	Students in the experimental group had better results than individuals.
[72]	Students in groups (three to four) learned programming by developing games	Authors seek to understood how students worked in groups and found that those who had planned their activity exhibit better results.
[73]	Programming videos streamed over Twitch and the interaction of audience was analyzed.	Live-streaming of programming activities was seen as positive on promoting programming collaboration.
[74]	Authors wanted to discover the impact and diference of three pedagogical resources: unplugged activity, Code.org, and a big touch display.	Students had different opinions about the technologies and no consensus was found. It may suggest a possibility to explore multiple technologies during learning.
[75]	Python online IDE with support for communication and algorithm visualization	Based on the log usage of this platform, authors concluded that it supported students need for assistance.
[76]	E-mail was used to delivery assignments and support collaboration between students.	Students evaluated the platform as useful.
[77]	Tangible and programmable object for students with impaired vision.	Authors evaluated the platform as a useful way to promote collaborative learning in this specific context.
[78]	Learning analytics techniques (LST) were used to identify students in need of assistance during a collaborative programming activity.	Experimental group presented better engagement and learning outcomes.
[79]	Use of automatic group recommendation strategy	Students who were grouped by the platform presented better results than solo student's.
[80]	Mobile application that tracks user actions for automatic group formation.	Students who were grouped automatically had better performance than manually group formation.
[81]	Students in groups created musics using tangible object based on programming concepts.	Authors concluded that the object successfully supported collaboration, however computational concepts were less explored.
[82]	A Remote robotic laboratory supported group programming activities.	Students were satisfied using the platform and had better academic achievement, than those who did not use it.
[83]	A platform for sharing annotations in Scratch exercises.	Creating and revising annotations was positively correlated with learning, however, peer revising did not present a statistically significant correlation.
[84]	Pair-programming platform with gamification characteristics.	Students who used the platform presented better planning and collaboration skills than those who did not use it.
[85]	Gamified CSCL platform	Students who used the platform changed positively their collaboration and problem-solving skills.
[86]	Automatic pair formation and content personalization.	Students positively rated the platform and the content recommendation.
[87]	Real-time feedback displayed to the students while they were collaborating and group scaffolding.	The intervention did not show a clear effect on collaboration, however, group scaffolding helped participants in paying attention to their collaborative behavior.

A. Quality analysis

The adequacy of the study to the criteria defined in Section III-B are presented in Table III.

TABLE III
QUALITY INDICATORS (Y - YES, N - NO, P - PARTIALLY, N/A - NOT APPLICABLE).

Study	Q1	Q2	Q3	Q4	Q5	Q6	Q7
[17]	Y	P	Y	Y	N	Y	Y
[72]	Y	P	Y	N/A	N/A	N/A	N/A
[73]	N	P	N	N/A	N/A	N/A	N/A
[75]	Y	N	P	N/A	N/A	N/A	N/A
[68]	Y	P	P	Y	N	N	Y
[15]	N	P	P	Y	P	Y	Y
[69]	Y	Y	Y	N	N	N	N
[18]	N	N	Y	N	N	Y	Y
[14]	N	P	Y	N	N	Y	N
[12]	Y	Y	Y	Y	N	Y	Y
[70]	Y	N	P	Y	N	N	Y
[13]	Y	P	N	P	N	Y	Y
[16]	Y	Y	Y	P	N	Y	Y
[71]	N	Y	P	N	N	Y	Y
[74]	N	Y	Y	N	N	N	Y
[76]	N	P	Y	N	N	N	N
[77]	Y	Y	Y	N	N	N	N
[78]	Y	Y	P	Y	N	Y	Y
[79]	Y	Y	N	P	N	Y	N
[80]	Y	Y	Y	Y	N	Y	N
[81]	Y	Y	Y	N	N	N	Y
[82]	Y	Y	Y	N	N	N	Y
[83]	N	P	Y	Y	N	Y	N
[84]	Y	Y	Y	Y	N	N	Y
[85]	Y	P	N	N	N	N	N
[86]	N	N	N	N	N	Y	N
[87]	N	P	Y	Y	N	Y	Y

Surprisingly, 48.15% of studies did not mention any details about the characteristics of the sample or described it partially. The absence of this information compromises the analysis of the results, as the context of the study is of fundamental importance for scientific purposes [11].

Other threats to the validity of experiments were observed. For example, only one of the experimental studies provided enough detail about the design of the experiment (factors, levels, execution, selection of the sample, among others). Also, 41.6% did not use pre/post-test design and 37.5% did not use the control-group. Lastly, only two studies used standardized tests.

As a consequence, replicating experiments and/or meta-analysis conduction are impaired. [3]. Due to the aforementioned problems, a statistical meta-analysis process could not be performed, and it is not possible to compute an overall effect-size for CSCL in programming education [88]. Similar problems regarding experimental design failures were found by other systematic reviews in programming education [89]; [90]. Authors of these studies found a small number of experimental studies available on scientific literature, and methodological failures related to the control-group definition and the lack of standardized tests.

A special comment must be made on the use of control-groups in the reviewed studies. They used two types: i) students who worked individually [71]; [79]; and ii) students who worked collaboratively, without the intervention [13]; [70]; [17]; [68]; [18]; [12]; [13]; [78]; [84]. A control-group with students studying individually can isolate the influence of collaboration on the dependent variable. Still, it is not possible to establish if the variation on outcomes is a natural consequence of collaboration or if it resulted from the collaboration using the CSCL. A possible solution is to use a control-group with students working on groups without the intervention. However, it is suggested for future studies to use a mix of these approaches to better isolate the benefits of collaboration, especially in CSCL.

B. Paper analysis

Results will be presented following the order of research questions.

1) What collaborative resources were used?: A resource is defined as a feature in the software or a strategy, used to support collaboration. Nineteen different resources were identified and are listed in Table IV. A single CSCL may have used one or multiple resources presented in this list, with a mean of 3.25 per study.

These resources were categorized according to their purpose (Table V). A single resource may belong to one or multiples categories.

Collaborative programming editor was the most used resource. Usually, they included built-in programming exercises, real-time code modifications and mouse pointer visualization (a student could see where their peers' mouse were pointing). This result supports the findings of [7], where collaborative algorithm writing was also the most frequently collaborative resource. It should be noted that this study considered situations with and without technology.

Other techniques were also adopted, for example, collaborative algorithm visualization [75], support for pair-programming (e.g.: automatic role selection - the system defines who will be the programmer and the reviewer, with dynamic switching) [68], and the possibility to request help from other students.

The aforementioned examples support the constructs of pedagogical theories like Socio-Constructivism and Social Cognitive Theory [91]. Although most studies did not mention the theoretical approach that motivated their decisions.

Communication between students was mainly supported by textual chat. However, as programming activities are also textual, we argue that this type of communication might compromise collaboration. To mitigate this problem, microphone-support could be used, as suggested by [85]. Studies also leveraged existing resources in Universities. E-mail, Moodle Plugins and Wiki platforms were created/used to support the CSCL practice in programming education [69]; [12]; [17]. While benefiting from the existing technologies, these non-specific programming environments might impose barriers to the learning process, causing dissatisfaction in students, as

TABLE IV
COLLABORATIVE RESOURCES USED IN CSCL.

Resource	Description	Studies	Count
Collaborative programming editor	Algorithm editor is shared in real-time between students.	[68]; [69]; [12]; [70]; [75]; [78]; [82]; [84]; [85];	9
Scaffolding for problem-discussion	Use of scaffolds to stimulate student's share of ideas and thoughts about programming questions	[17]; [15] [69]; [18]; [14]; [12]; [72]; [77];	8
Peer review	Students review algorithms made by peers	[17]; [14]; [12]; [16]; [76]; [78]; [83]; [86]	8
Chat	Communication between students, mainly by text.	[68]; [14]; [70]; [73]; [75]; [82]; [85]	7
Sharing technological resource	CSCL artifacts (computer, physical object, among others) are shared between students	[13]; [16]; [71]; [72]; [74]; [77]; [81]; [87]	8
Pair-programming support	CSCL platform provides functionalities for students to alternate their roles (programmer and reviewer) while programming	[68]; [17]; [71]; [74]; [84];	5
Physical artifacts	Use of tangible objects to stimulate collaboration	[18]; [16]; [77]; [81]	4
Discussion forums	Student can post questions or help other students	[17]; [14] [69];	3
Group task assignment	For each student in a group a predefined task is assigned	[68]; [18]; [79]	3
Individual contribution awareness	The progress for each student in a group is shared between members	[85]; [87]	2
Automatic group formation	Student's group creation is automatic and used to stimulate interaction and knowledge sharing	[80]; [79]	2
Programming observation	A student writes a solution to an algorithm problem while another observes it	[73]; [75]	2
Group goals definition	The group receives an explicit list of learning goals	[82]; [85]	2
Gamification	Use of gamification techniques to stimulate collaboration	[85]; [84]	2
Algorithm sharing	A student can share their solution to a programming task with other students	[70];	1
Project-Based Learning	Students engage in real-world problems	[16];	1
Task recommendation	Automated task/exercise selection	[79]	1
Help request	Students work individually, but can ask for help from one of its peers	[80];	1
Voting-system	Each group member can propose a solution that will be voted by its peers	[15];	1

TABLE V
CATEGORIZATION OF RESOURCES.

Category	Description	Resources
Programming-oriented	Used to support collaborative programming activities	Collaborative programming editor; Peer-review; Pair-programming support; Physical artifacts; Programming observation; Algorithm sharing; Help request
Pedagogical	Techniques and strategies to guide the educational process	Scaffolding for problem-discussion; Group task assignment; Task recommendation; Project-Based Learning;
Self-regulation	Scaffolding to support students regulatory skills	Individual contribution awareness; Group goals definition; Help request
Group support	Group activities orchestration	Automatic group formation; Group goals definition; Voting-system
Communication	Resources to mediate communication between students	Chat; Discussion forums
Collaborative stimulation	Encouraging collaboration between students	Sharing technological resource; Physical artifacts;
Motivational	Stimulating students motivation	Physical artifacts; Gamification

related in [17]. Further investigation is needed to evaluate the impact of these platforms in the context of collaborative programming education.

The use of mobile devices has also been investigated [15]; [18]; [80], but to a less extent. Despite the opportunity for ubiquitous learning (anytime and anywhere), these studies only used the device in conjunction with classroom lectures. For future studies, we recommend exploring the pervasive possibilities of collaborative programming learning.

Despite the importance of Learning Analytics (LA) to CSCL [92], only three studies (11.1%) used this technique. They were applied in automatic group formation [80]; [79], and to assist instructors in identifying students in need of assistance [78].

Researchers were also interested in alternatives forms of programming education. For example, the use of visual programming languages (e.g. Code.org) [74], storytelling platforms (Scratch) [16] and tangible objects were explored, especially to promote engagement with children. Scratch platform is an interesting example, where collaboration does not happen during the code development, but through communities of practice. In this context, users can share their projects with the community, and this can raise feedback or support other students.

Tangible objects also deserve attention, due to their potential in promoting student engagement [77]; [81], as this is an important aspect for collaborative environments [92]. However, further investigation is needed to establish the impacts on programming learning, as the actual results could not support it.

2) Which features are most effective?: The use of CSCL in programming education is supported by results from experimental studies. Only three works did not found differences on the use of CSCL, compared to the control group, without this technology [69]; [70]; [87], and only one related negative results [17].

However, this conclusion must be interpreted with caution, as the majority of experimental settings had inconsistencies that may impact the results. For example, the absence of a control group, sample sizes, selection of variables, lack of standardized tests, among others.

Furthermore, it is still not possible to answer which resource produce positive, or even negative effects on students outcome. Studies assessed intervention as a whole, without isolating the individual impact of resources. Considering the trade-off between including features on software vs time to development and cognitive load [93]), it is of special importance to gain knowledge over what features are effective, and which are not.

3) What were the outcomes measured in the experiments?: A variety of outcomes were measured in the experimental analysis (Table VI). Studies mostly focused on programming learning outcomes (n=18), with only one study using a standardized test [14]. Researchers were also interested in the perception of students over the platform. Again, only one study [82] used a standardized metric.

Despite being an important issue for education, only two studies explored motivational aspects [85]; [84]. Both used gamification strategies and achieved positive results.

The rest of the measurements were generally specific to the research context.

4) How collaboration is structured?: Two formats of student collaboration were identified: a) pair-programming (n=13) and b) programming in groups with more than two students (n=14). Pair-programming is a known technique in software development and was replicated in programming education, in two arrangements. The first is a cooperative format, where one student with a higher knowledge guides the format. This format resembles what Vygotsky called Zone of Proximal Development [91].

The second arrangement was collaborative, in which students work together to solve a programming task. Usually, students were designed as drivers or navigators, where the first was responsible for writing code, while the other for revision, changing their roles over time.

In the context of pair-programming, researchers investigated: a) how students collaborate [75]; [17]; [68]; [71]; [74]; [76]; [77] b) the use of a pre-defined set of steps (called script) to guide the collaboration [68]; and c) automatic pair formation [80];

In group programming, students worked together in the same programming tasks and two formats were identified: i) jigsaw-format, wherein a bigger problem is split into small parts assigned to each student; and ii) students defined how they will approach the problem and distribute the tasks.

TABLE VI
OUTCOMES MEASURED IN EXPERIMENTS.

Outcome	Description	Studies	Count
Programming knowledge	Knowledge in programming topics and/or passing in a course	[17]; [68]; [15]; [69]; [18]; [14]; [12]; [13]; [16]; [71]; [76]; [77]; [78]; [79]; [81]; [82]; [83]; [86]	18
User-acceptance	Usability and/or questions related to the acceptance of the system	[13]; [74]; [82]; [83]; [84]. [77];	6
Specific metrics	Context-specific measurements	[80]; [81]	2
Number of defects	Number of defects detected in algorithm	[17]; [70];	2
Creativity index	Specific metrics to measure how the proposed solution is considered creative.	[72]; [73]	2
Algorithm complexity	Specific metric to measure the complexity of algorithms (not related to O complexity)	[71]; [72]	2
Code correctness	An expert analysis of the algorithm correctness	[71]; [72]	2
Motivation	Student motivation	[15];	1
Logical tests	Tests to measure logical skills	[14];	1
Self-efficacy	Student's self-efficacy	[12]	1
Engagement	Student engagement during platform use	[77];	1
Self-regulated learning	Student self-regulated learning skills	[78];	1

Collaboration also can be classified by the moment it happened: a) synchronous, b) asynchronous or c) hybrid-format. In the first case, students collaborated in real-time to solve programming tasks. The characteristics of introductory programming education, in which students usually work on small tasks, each lasting a few minutes, might be an explanation for the use of synchronous tasks.

In asynchronous collaboration, students did not need to be connected to the platform at the same time. It was mainly used for task discussion and chat. Still, four studies proposed programming exercises asynchronously. The use of problem-based learning can be employed in this scenario [12]. Lastly, in the hybrid format, studies used both synchronous and asynchronous features.

A recurrent challenge for CSCL, including in programming education, is to promote student's engagement in group activities [68]. To mitigate this problem, [79] and [80] explored the use of learning analytics techniques to adapt the activities and group formation. Results were satisfactory when compared to the control group.

5) How collaborative work was measured?: In CSCL environments it is essential to understand how the collaborative process occurs [92]. Still, 37% of studies did not carry out any type of analysis on this [15]; [18]; [14]; [70]; [13]; [16];

TABLE VII
COLLABORATION MEASUREMENTS TECHNIQUES.

Outcome	Description	Studies	Count
Analysis of how students interacted with the platform	System's logs or observation on the use of the platform	[17]; [68]; [72]; [75]; [76]; [83]; [84]; [85]; [86]	9
Interviews	Students were interviewed individualy or in groups	[17]; [69]; [73]; [74]. [84]; [85]; [86]; [87]; [76]	9
Dialog analysis and group interaction observation	Analyzes the interaction between peers in a group.	[72]; [74]; [81]; [77]; [87]	5
Peer evaluation	Students evaluated their pairs	[68]; [69];	2
Number of algorithm lines	Count the number of lines written by students	[68]	1
Group self-efficacy	Evaluate how groups self-efficacy was affected by the intervention.	[12]	1
Expert evaluation	Experts evaluated the collaborative features of the platform.	[82]	1
Social Network Analysis	Create a graph of interaction between students.	[85]	1
Multimodal data	Use of sensors, like eye-tracking and Xbox kinetic.	[87]	1

[71]; [78]; [79]; [80].

Multiple measurements were adopted to comprehend how collaboration impacted students, how it occurred and how it was affected by the intervention. Studies used multiple instruments, and in some cases mixing qualitative and quantitative approaches (Table VII).

Important variables for education received less importance, for example, motivation (n=1), self-efficacy (n=1) and engagement (n=1) [94]. It is relevant for future studies to focus on stimulating and measuring these metrics.

Only six studies used data generated by students to analyze how they collaborate [75]; [85]; [72]; [68]; [84]; [17]. For example, click-streams, counters on the access to learning materials, screencasts to analyze student's behavior, logs from chat, among others were collected. There is room to explore this area in future research.

Some studies presented a superficial analysis, for example, with closed-ended questions that sought to capture the student's preferences on working collaboratively or individually, and their perception of the usefulness of the collaborative support [74]; [76]. Although this information is valuable, it does not provide meaningful details on the interactions involved in collaborative programming learning.

Based on the results, we can concluded that the majority of reviewed studies focused on providing an intervention to stimulate a collaborative programming task, leaving the investigation of how students collaborate during programming behind. However, this lack of understanding may impact the development of the field [95].

Only [72]; [75]; [73]; [77]; [81]; [17]; [68]; [84]; [73]; [85] provided details information about the collaborative process. The focus of analysis was mainly on pair collaboration, and some of the findings are presented below.

Lin et. al, observed that students who paired face-to-face felt more comfortable, less stressed and more patient when compared to the experimental group that used a CSCL, also in pairs [17]. This raises an important question on the benefits of CSCL when compared to face-to-face collaboration. Also, they found that some students did not see the benefits of collaboration in improving their programming skills, and in worse situations, found that the collaboration slowed down their progress. Still not clear how these perceptions impact the learning.

Negative consequences of the collaboration were also identified in [84]. Students were working in pairs, switching their roles as drivers and navigators (who assist the programmer, making suggestions and point their errors). For some, the role of navigator gave them the feeling that they contributed less. Moreover, there were reports that some students did not know how to perform this role. Finally, some students reported that even working in pairs, they would rather develop their solutions on their own than having to discuss them with their peers. These results might be an indicator of a necessity to explicitly guide the collaboration, seeking to stimulate the interaction between students and explaining their role objectives and the importance of collaboration to the learning process.

Lastly, [75] analyzed chat logs and found that students use collaborative resources when they felt in need of assistance. While, Tsompanoudi et al., had found that pairing students with similar skills produced better results and stimulates balanced cooperation [68].

Still, the understanding of how students collaborate on programming learning (with and without CSCL) deserves more attention. A combination of quantitative and qualitative analysis, accompanied by the use of user-generated data and techniques like grounded-theory, may advance the understanding of the collaborative process involved in programming education.

V. CONCLUSION

Computer-supported collaborative learning has been used for years in programming education. However, there was a lack of studies summarizing the findings of this area. A systematic literature review was conducted in this study to construct this body of knowledge.

Twenty-seven studies were analyzed to identify how collaboration was stimulated, evaluated and structured, to promote programming learning. In these, nineteen strategies were used to support collaboration, twelve outcomes were measured and ten collaborative measurements were made.

Results are relevant to understand the use of CSCL in programming education. Still, some of our research questions were not fully answered due to experimental design problems found on analyzed studies. These concerns are discussed and proposals are made to mitigate them.

Future studies must pay attention to the design of the experiment, especially on the use of standardized tests to favor between-study comparisons in a meta-analysis process. Also, features used to promote collaboration must be evaluated isolated to identify which one is most effective.

References

[1] R. P. Medeiros, G. L. Ramalho, and T. P. Falcão, “A systematic literature review on teaching and learning introductory programming in higher education,” *IEEE Transactions on Education*, vol. 62, no. 2, pp. 77–90, 2018.

[2] C. Watson and F. W. Li, “Failure rates in introductory programming revisited,” in *Proceedings of the 2014 conference on Innovation & technology in computer science education*. ACM, 2014, pp. 39–44.

[3] B. R. Belland, A. E. Walker, N. J. Kim, and M. Lefler, “Synthesizing results from empirical research on computer-based scaffolding in stem education: A meta-analysis,” *Review of Educational Research*, vol. 87, no. 2, pp. 309–344, 2017.

[4] L. Lipponen, “Exploring foundations for computer-supported collaborative learning,” in *Proceedings of the conference on computer support for collaborative learning: Foundations for a CSCL community*. International Society of the Learning Sciences, 2002, pp. 72–81.

[5] H. Jeong, C. E. Hmelo-Silver, and K. Jo, “Ten years of computer-supported collaborative learning: A meta-analysis of cscl in stem education during 2005-2014,” *Educational Research Review*, p. 100284, 2019.

[6] J. Hattie, *Visible learning for teachers: Maximizing impact on learning*. Routledge, 2012.

[7] O. Revelo-Sánchez, C. A. Collazos-Ordóñez, and J. A. Jiménez-Toledo, “Collaborative work as a didactic strategy for teaching/learning programming: a systematic literature review,” *TecnoLógicas*, vol. 21, no. 41, pp. 115–134, 2018.

[8] A. Knutas, J. Ikonen, and J. Porras, “Computer-supported collaborative learning in software engineering education: a systematic mapping study,” *International Journal on Information Technologies Security*, 2014.

[9] B. Kitchenham, “Procedures for performing systematic reviews,” *Keele, UK, Keele University*, vol. 33, no. 2004, pp. 1–26, 2004.

[10] A. for Computing Machinery (ACM), “Computer science curricula 2013: Curriculum guidelines for undergraduate degree programs in computer science,” 2013.

[11] R. Jain, *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. John Wiley & Sons, 1990.

[12] X.-M. Wang and G.-J. Hwang, “A problem posing-based practicing strategy for facilitating students’ computer programming skills in the team-based learning mode,” *Educational Technology Research and Development*, vol. 65, no. 6, pp. 1655–1671, 2017.

[13] N. Ideris, S. M. Baharudin, and N. Hamzah, “The effectiveness of scratch in collaborative learning on higher-order thinking skills in programming subject among year-six students,” in *4th ASEAN Conference on Psychology, Counselling, and Humanities (ACPCH 2018)*. Atlantis Press, 2019.

[14] M. Othman, N. M. Zain, U. H. Mazlan, and R. Zainordin, “Assessing cognitive enhancements in introductory programming through online collaborative learning system,” in *2015 International Symposium on Mathematical Sciences and Computing Research (iSMSC)*. IEEE, 2015, pp. 7–12.

[15] L. M. Serrano-Cámara, M. Paredes-Velasco, J. Á. Velázquez-Iturbide, C.-M. Alcover, and M. E. Castellanos, “Mocas: A mobile collaborative tool for learning scope of identifiers in programming courses,” 2016.

[16] S. Papavlasopoulou, K. Sharma, M. Giannakos, and L. Jaccheri, “Using eye-tracking to unveil differences between kids and teens in coding activities,” in *Proceedings of the 2017 Conference on Interaction Design and Children*. ACM, 2017, pp. 171–181.

[17] Y.-T. Lin, C.-C. Wu, and C.-F. Chiu, “The use of wiki in teaching programming: Effects upon achievement, attitudes, and collaborative programming behaviors,” *International Journal of Distance Education Technologies (IJDET)*, vol. 16, no. 3, pp. 18–45, 2018.

[18] J. A. J. Toledo, C. A. Collazos, M. O. Cantero, and M. Á. Redondo, “Collaborative strategy with augmented reality for the development of algorithmic thinking,” in *Iberoamerican Workshop on Human-Computer Interaction*. Springer, 2018, pp. 70–82.

[19] S. Xinogalos, C. Malliarakis, D. Tsompanoudi, and M. Satratzemi, “Microworlds, games and collaboration: three effective approaches to support novices in learning programming,” in *Proceedings of the 7th Balkan Conference on Informatics Conference*. ACM, 2015, p. 39.

[20] Y.-L. Lin, M.-W. Lee, and I.-H. Hsiao, “An exploratory study on programming orchestration technology.” in *PACIS*, 2018, p. 155.

[21] Z. Sun, Z. Li, S. Zaorski, T. Nishimori, T. Maesako, M. Nakamura, and R. Imamura, “A documentation platform for supporting and assessing collaborative knowledge building in learning computer programming,” vol. 20, pp. 77–89, 2015.

[22] J. Tvarozek and P. Jurkovic, “Student-generated content improves online learning of programming,” *International Journal of Human Capital and Information Technology Professionals (IJHCITP)*, vol. 7, no. 4, pp. 79–92, 2016.

[23] M. Villamor, Y. V. Paredes, J. D. Samaco, J. F. Cortez, J. Martinez, and M. M. Rodrigo, “Assessing the collaboration quality in the pair program tracing and debugging eye-tracking experiment,” in *International Conference on Artificial Intelligence in Education*. Springer, 2017, pp. 574–577.

[24] I. Musabirov, P. Okopny, and S. Pozdniakov, “Enabling information access in virtual learning environment: The case of data science minor,” in *Proceedings of the 9th Nordic Conference on Human-Computer Interaction*. ACM, 2016, p. 119.

[25] D. Weintrop, D. Bau, and U. Wilensky, “The cloud is the limit: A case study of programming on the web, with the web,” *International Journal of Child-Computer Interaction*, vol. 20, pp. 1–8, 2019.

[26] P. L. da R Rodrigues, L. P. Franz, J. F. P. Cheiran, J. P. S. da Silva, and A. S. Bordin, “Coding dojo as a transforming practice in collaborative learning of programming: an experience report,” in *Proceedings of the 31st Brazilian Symposium on Software Engineering*. ACM, 2017, pp. 348–357.

[27] G.-m. Chen, “Programming language teaching model based on computational thinking and problem-based learning,” in *2017 2nd International Seminar on Education Innovation and Economic Management (SEIEM 2017)*. Atlantis Press, 2017.

[28] V. Kumar, T. Somasundaram, S. Harris, D. Boulanger, J. Seanosky, G. Paulmani, K. Panneerselvam *et al.*, “An approach to measure coding competency evolution,” in *Smart Learning Environments*. Springer, 2015, pp. 27–43.

[29] T. Daradoumis, J. M. M. Puig, M. Arguedas, and L. C. Liñan, “Analyzing students’ perceptions to improve the design of an automated assessment tool in online distributed programming,” *Computers & Education*, vol. 128, pp. 159–170, 2019.

[30] A. Luxton-Reilly, A. Lewis, and B. Plimmer, “Comparing sequential and parallel code review techniques for formative feedback,” in *Proceedings of the 20th Australasian Computing Education Conference*. ACM, 2018, pp. 45–52.

[31] D. Hegab, “Modernizing women’s learning in software development: A study on constructionist pedagogy and networked support,” Ph.D. dissertation, UC Irvine, 2015.

[32] O. H. Lu, A. Y. Huang, J. C. Huang, C. S. Huang, and S. J. Yang, “Early-stage engagement: Applying big data analytics on collaborative learning environment for measuring learners’ engagement rate,” in *2016 International Conference on Educational Innovation through Technology (EITT)*. IEEE, 2016, pp. 106–110.

[33] M. Othman, S. H. Q. Alias, and N. F. M. Rashidi, “Enhanced collaborative e-learning for programming using open learner model,” *Computing Research & Innovation (CRINN), Vol. 1, November 2016*, p. 64, 2016.

[34] S. Svetsky and O. Moravcik, “The development of personalized educational software and cscl approach within the stem education,” in *2017 7th World Engineering Education Forum (WEEF)*. IEEE, 2017, pp. 823–828.

[35] I. Anyfanti, M. Z. KonstantinosVasileiadis, A. Vgenopoulou *et al.*, “Computer supported collaborative learning in small teams for scratch: Programming skills for year 4 students at the 6th primary school of patras, greece,” *Transforming Schools into Innovative Learning Organisations*, 2015.

[36] M. Bernard and E. Bachu, “Enhancing the metacognitive skill of novice programmers through collaborative learning,” in *Metacognition: Fundaments, applications, and trends*. Springer, 2015, pp. 277–298.

[37] M. H. Pietruchinski and A. R. Pimentel, "An architectural model of multi-agent systems for student evaluation in collaborative game software," *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 3806, p. 10003001, 2015.

[38] I.-H. Hsiao and F. Naveed, "Identifying learning-inductive content in programming discussion forums," in *2015 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2015, pp. 1–8.

[39] J. Zhu, J. Warner, M. Gordon, J. White, R. Zanelatto, and P. J. Guo, "Toward a domain-specific visual discussion forum for learning computer programming: An empirical study of a popular mooc forum," in *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2015, pp. 101–109.

[40] A. I. Molina, J. Gallardo, M. Á. Redondo, and C. Bravo, "Assessing the awareness mechanisms of a collaborative programming support system," *Dyna*, vol. 82, no. 193, pp. 212–222, 2015.

[41] W. Xing, "Large-scale path modeling of remixing to computational thinking," *Interactive Learning Environments*, pp. 1–14, 2019.

[42] C. Brady, D. Weintrop, K. Gracey, G. Anton, and U. Wilensky, "The ccl-parallax programmable badge: Learning with low-cost, communicative wearable computers," in *Proceedings of the 16th Annual Conference on Information Technology Education*. ACM, 2015, pp. 139–144.

[43] K. J. Reyes, A. Saavedra, C. A. Collazos, and J. A. Hurtado, "Diseñando actividades colaborativas y lúdicas para la enseñanza de la programación en equipos conformados por jóvenes," *Revista Colombiana de Computación*, vol. 17, no. 2, pp. 9–22, 2016.

[44] D. Shawky and A. Badawi, "Towards a personalized learning experience using reinforcement learning," in *Machine Learning Paradigms: Theory and Application*. Springer, 2019, pp. 169–187.

[45] N. G. Monjelat and P. S. San Martín, "Programar con scratch en contextos educativos:¿ asimilar directrices o co-construir tecnologías para la inclusión social?" *Praxis*, 2016.

[46] N. Moumoutzis, G. Boukeas, V. Vassilakis, N. Pappas, C. Xanthaki, I. Maragkoudakis, A. Deligiannakis, and S. Christodoulakis, "Design, implementation and evaluation of a computer science teacher training programme for learning and teaching of python inside and outside school," in *Interactive Mobile Communication, Technologies and Learning*. Springer, 2017, pp. 575–586.

[47] F. Jurado and M. A. Redondo, "Ims-lti and web-services for integrating moodle to an eclipse-based distributed environment for learning to program," *International Journal of Engineering Education*, vol. 32, no. 2, pp. 1007–1014, 2016.

[48] J. Moreno-León, G. Robles, and M. Román-González, "Examining the relationship between socialization and improved software development skills in the scratch code learning environment." *J. UCS*, vol. 22, no. 12, pp. 1533–1557, 2016.

[49] L. D. P. Machado, C. D. M. Berckenbrock, and I. Z. Siple, "Desenvolvimento de aplicativos para aprendizagem colaborativa apoiada por dispositivos móveis: uma análise dos requisitos," *Anais do Computer on the Beach*, pp. 001–010, 2017.

[50] G. U. Nneji, J. Deng, S. S. Shakher, H. N. Monday, B. C. Mbonu, and A. Ogungbile, "A collaborative learning approach for integrated time based online environment," in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. IEEE, 2018, pp. 1138–1144.

[51] Z. Sun, R. Liu, L. Luo, M. Wu, and C. Shi, "Exploring collaborative learning effect in blended learning environments," *Journal of Computer Assisted Learning*, vol. 33, no. 6, pp. 575–587, 2017.

[52] S.-L. Wang and H.-T. Hong, "The roles of collective task value and collaborative behaviors in collaborative performance through collaborative creation in cscl," *Educational Technology Research and Development*, vol. 66, no. 4, pp. 937–953, 2018.

[53] M. Sáiz-Manzanares and R. Marticorena Sánchez, "Aprendizaje colaborativo y auto-regulación en aprendices noveles," 2016.

[54] G. Palaigeorgiou and I. Kazanidis, "Wikis as a mediation platform for developing learning communities: The weki framework," in *International Symposium on Emerging Technologies for Education*. Springer, 2016, pp. 463–472.

[55] A. Weibert, M. Mouratidis, R. Khateb, S. Rüller, M. Hosak, S. Potka, K. Aal, and V. Wulf, "Creating environmental awareness with upcycling making activities: A study of children in germany and palestine," in *Proceedings of the 2017 Conference on Interaction Design and Children*. ACM, 2017, pp. 286–291.

[56] D. A. Fields, Y. B. Kafai, and M. T. Giang, "Coding by choice: A transitional analysis of social participation patterns and programming contributions in the online scratch community," in *Mass collaboration and education*. Springer, 2016, pp. 209–240.

[57] K. Øygardslia, "Students as game designers: Exploring collaborative game-based learning activities in the classroom," Ph.D. dissertation, 2018.

[58] R. Zatarain-Cabada, M. L. Barrón-Estrada, F. G. Hernández, and G. Alor-Hernandez, "Evaluando afecto en un entorno de aprendizaje para java." *Research in Computing Science*, vol. 111, pp. 123–133, 2016.

[59] V. Nguyen, H. H. Dang, N.-K. Do, and D.-T. Tran, "Enhancing team collaboration through integrating social interactions in a web-based development environment," *Computer Applications in Engineering Education*, vol. 24, no. 4, pp. 529–545, 2016.

[60] J. Zhang, "An investigation of technology design features for supporting real-time collaborative programming in an educational environment," Ph.D. dissertation, 2018.

[61] A. A. Moh'd Al-Jarrah, "Collaborative virtual environments for introductory programming (cveip)," Ph.D. dissertation, New Mexico State University, 2016.

[62] L. H. LeGault, "Pair programming and unobtrusive monitoring: Toward an automated partner matching system," Ph.D. dissertation, 2018.

[63] I.-H. Hsiao and Y.-L. Lin, "Enriching programming content semantics: An evaluation of visual analytics approach," *Computers in Human Behavior*, vol. 72, pp. 771–782, 2017.

[64] M. M. Villamor and M. T. Rodrigo, "Characterizing collaboration in the pair program tracing and debugging eye-tracking experiment: A preliminary analysis." *International Educational Data Mining Society*, 2017.

[65] J. M. Reilly, M. Ravenell, and B. Schneider, "Exploring collaboration using motion sensors and multi-modal learning analytics." *International Educational Data Mining Society*, 2018.

[66] D. Tsompanoudi, M. Satratzemi, and S. Xinogalos, "Distributed pair programming using collaboration scripts: An educational system and initial results." *Informatics in Education*, vol. 14, no. 2, pp. 291–314, 2015.

[67] B. A. Kos, "The collaborative learning framework: Scaffolding for untrained peer-to-peer collaboration," 2017.

[68] D. Tsompanoudi, M. Satratzemi, and S. Xinogalos, "Evaluating the effects of scripted distributed pair programming on student performance and participation," *IEEE Transactions on education*, vol. 59, no. 1, pp. 24–31, 2015.

[69] E. Lovos, "Ambiente de desarrollo virtual para el aprendizaje de la programación: un estudio de caso en la lic. de sistemas de la universidad nacional de río negro, patagonia argentina/virtual development environment for learning programming: a case study at bachelor systems of rio negro university, patagonia argentina," *Revista Internacional de Aprendizaje en Ciencia, Matemáticas y Tecnología*, vol. 2, no. 1, 2015.

[70] J. P. Ucan, O. S. Gomez, and R. A. Aguilar, "Assessment of software defect detection efficiency and cost through an intelligent collaborative virtual environment," *IEEE Latin America Transactions*, vol. 14, no. 7, pp. 3364–3369, 2016.

[71] S. Papadakis, "Is pair programming more effective than solo programming for secondary education novice programmers?: A case study," *International Journal of Web-Based Learning and Teaching Technologies (IJWLTT)*, vol. 13, no. 1, pp. 1–16, 2018.

[72] B. Wu, Y. Hu, A. Ruis, and M. Wang, "Analysing computational thinking in collaborative programming: A quantitative ethnography approach," *Journal of Computer Assisted Learning*, vol. 35, no. 3, pp. 421–434, 2019.

[73] T. Faas, L. Dombrowski, A. Young, and A. D. Miller, "Watch me code: Programming mentorship communities on twitch. tv," *Proceedings of the ACM on Human-Computer Interaction*, vol. 2, no. CSCW, p. 50, 2018.

[74] N. S. Molina-Moreno, M. S. Avila-Garcia, M. Bianchetti, D. Claudio-Gonzalez, and M. Pantoja-Flores, "Impact of technology in collaborative and interactive programming activities: Gathering children's feedback," in *2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT)*. IEEE, 2017, pp. 179–183.

[75] P. J. Guo, J. White, and R. Zanelatto, "Codechella: Multi-user program visualizations for real-time tutoring and collaborative learning," in *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2015, pp. 79–87.

[76] A. Neznanov and O. Maksimenkova, "The pasca: A mail based randomized blinded peer assessment system for complex artifacts," *Procedia Computer Science*, vol. 96, pp. 826–837, 2016.

[77] A. Thieme, C. Morrison, N. Villar, M. Grayson, and S. Lindley, "Enabling collaboration in learning computer programing inclusive of children with vision impairments," in *Proceedings of the 2017 Conference on Designing Interactive Systems*. ACM, 2017, pp. 739–752.

[78] O. H. Lu, J. C. Huang, A. Y. Huang, and S. J. Yang, "Applying learning analytics for improving students engagement and learning outcomes in an moocs enabled collaborative programming course," *Interactive Learning Environments*, vol. 25, no. 2, pp. 220–234, 2017.

[79] M. Kompan and M. Bielikova, "Enhancing existing e-learning systems by single and group recommendations," *International Journal of Continuing Engineering Education and Life Long Learning*, vol. 26, no. 4, pp. 386–404, 2016.

[80] M. Berland, D. Davis, and C. P. Smith, "Amoeba: Designing for collaboration in computer science classrooms through live learning analytics," *International Journal of Computer-Supported Collaborative Learning*, vol. 10, no. 4, pp. 425–447, 2015.

[81] A. Xambó, B. Drozda, A. Weisling, B. Magerko, M. Huet, T. Gasque, and J. Freeman, "Experience and ownership with a tangible computational music installation for informal learning," in *Proceedings of the Eleventh International Conference on Tangible, Embedded, and Embodied Interaction*. ACM, 2017, pp. 351–360.

[82] M. S. d. S. Lopes, "Ambiente colaborativo para ensino aprendizagem de programação integrando laboratório remoto de robótica," Ph.D. dissertation, Universidade Federal da Bahia, 2017.

[83] A. Y. Su, C. S. Huang, S. J. Yang, T.-J. Ding, and Y. Hsieh, "Effects of annotations and homework on learning achievement: An empirical study of scratch programming pedagogy," *Journal of Educational Technology & Society*, vol. 18, no. 4, p. 331, 2015.

[84] J. Shi, A. Shah, G. Hedman, and E. O'Rourke, "Pyrus: Designing a collaborative programming game to promote problem solving behaviors," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 2019, p. 656.

[85] A. Knutas *et al.*, "Increasing beneficial interactions in a computer-supported collaborative environment," Ph.D. dissertation, Lappeenranta University of Technology, 2016.

[86] V. Bremgartner, J. F. Netto, and C. de Menezes, "Arcabouço conceitual de adaptação de recursos educacionais," in *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, vol. 7, no. 1, 2018, p. 1.

[87] E. L. Starr, J. M. Reilly, and B. Schneider, "Toward using multi-modal learning analytics to support and measure collaboration in co-located dyads." International Society of the Learning Sciences, Inc.[ISLS]., 2018.

[88] G. M. Sullivan and R. Feinn, "Using effect size—or why the p value is not enough," *Journal of graduate medical education*, vol. 4, no. 3, pp. 279–282, 2012.

[89] J. M. Costa and G. L. Miranda, "Relation between alice software and programming learning: A systematic review of the literature and meta-analysis," *British Journal of Educational Technology*, vol. 48, no. 6, pp. 1464–1474, 2017.

[90] J. Moreno-León and G. Robles, "Code to learn with scratch? a systematic literature review," in *2016 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 2016, pp. 150–156.

[91] P. Llave, P. Gruner, J. Kelly, and M. Tarantino, *Constructivism vs. Social Cognition Theory*. [Online]. Available: https://bit.ly/2SMDwoN

[92] G. Stahl, "A decade of cscl," *International Journal of Computer-Supported Collaborative Learning*, vol. 10, no. 4, pp. 337–344, 2015.

[93] P. A. Kirschner, "Cognitive load theory: Implications of cognitive load theory on the design of learning," *Learning and Instruction*, vol. 12, pp. 1–0, 2002.

[94] K. Li and J. M. Keller, "Use of the arcs model in education: A literature review," *Computers & Education*, vol. 122, pp. 54–62, 2018.

[95] P. Dillenbourg, S. Järvelä, and F. Fischer, "The evolution of research on computer-supported collaborative learning," in *Technology-enhanced learning*. Springer, 2009, pp. 3–19.