Journal of Computing in Higher Education
https://doi.org/10.1007/s12528-023-09356-3

Adopting distributed pair programming as an effective team learning activity: a systematic review

Fan Xu[1] · **Ana-Paula Correia**[2]

Accepted: 13 January 2023

Abstract

As online learning has become an inevitable trend in the post-peak era of the COVID-19 pandemic, distributed pair programming (DPP) is gaining momentum in both education and industry. DDP serves as a collaborative programming approach and also benefits the development of computational thinking, a fundamental skill in today's world. This study conducted a systematic review of studies on DPP published after 2010 to understand the themes and factors that impact the team effectiveness of DPP and thus inform future research and practices on how to better leverage this approach for teaching and learning. The results showed that individual characteristics attracted major investigations in the selected 23 studies, including prior programming experience, actual skill, perceived skill, gender, personality, time management, confidence, and self-esteem, with pair compatibility identified as a critical team design factor that significantly affects programmers' satisfaction. Although the feel-good factor in the team process was investigated, no significant impact was found. Under the team environment theme, we compared different opinions on the orientation (e.g., scripted roles) and the use of technology (e.g., integrated development environment tools). Future research should investigate how task structure influences team effectiveness of DPP and relates to computational thinking education. Additionally, because most studies were conducted in higher education contexts, more research in primary and secondary educational contexts is also needed.

Keywords Distributed pair programming · Systematic review · Computational thinking · Team learning · Collaborative learning · STEM education

✉ Fan Xu
xu.3849@osu.edu

[1] Department of Educational Studies, College of Education and Human Ecology, The Ohio State University, 1900 Kenny Rd., Columbus, OH 43210, USA

[2] Center On Education and Training for Employment, College of Education and Human Ecology, The Ohio State University, 1900 Kenny Rd., Columbus, OH 43210, USA

Published online: 15 February 2023

Introduction

The public health crisis that erupted in 2020 has brought devastating changes in education worldwide. According to UNESCO's report (2020), a large number of countries and regions experienced the disruption of learning at all levels of education due to the COVID-19 pandemic. To adapt to the pandemic, many schools and universities rapidly transitioned from traditional face-to-face or blended instruction to fully online learning solutions, named emergency remote teaching (Hodges et al., 2020). Researchers have argued that a potential influence of the COVID-19 crisis was making educators embrace the use of new technologies and promote educational innovations (Thomas & Rogers, 2020). Thus, educational institutions tend to rethink educational normalcy and develop learning strategies and approaches that are more sensitive to learners' needs in the post-pandemic era (Cahapay, 2020).

As online learning has become an inevitable trend, distributed pair programming (DPP) is playing an increasingly important role in programming education, through which two programmers work collaboratively on the same project from distributed locations under the support of technologies (Baheti et al., 2002). Remote work will likely increase tremendously in this decade due to the global health crises and programmers will continue to need to work in a geographically distributed manner. As a result, learning to program through the DPP approach from an early age is particularly necessary for preparing future programmers, developers, and engineers. On the other hand, computers and other technologies are becoming more ubiquitous in education, and computational thinking (CT) is often described as a fundamental 21st-century skill (Juškevičienė & Dagienė, 2018; Worrell et al., 2015). Programming has been defined as "the easiest and most appropriate ways to teach computational thinking" (Hsu et al., 2018, p.299), which is particularly true when programming is integrated into school or university curricula (Falkner et al., 2014; Kafai & Burke, 2014). And collaboration is an effective means for learners' mastery of CT patterns (Echeverría et al., 2019; Worrell et al., 2015; Zhong et al., 2016), as well as improvement in the overall problem-solving competency (Lin et al., 2016). Therefore, using DPP in programming education is beneficial to develop students' CT literacy in general and help them become independent thinkers and problem solvers.

Even though DPP benefits programming education from many aspects, these benefits will not occur spontaneously. Due to the open-ended, social, and complex nature of team-based learning activity, multiple factors could impact its effectiveness throughout the collaboration process (Faja, 2011; Zhong et al., 2016). There have been significant investigations into collocated pair programming; although limited exploration has been found on the geographical distribution in pair teaming and factors that may influence the effectiveness of DPP (Hanks et al., 2011). In addition, different empirical studies may show contradictory views on the impact of each factor. Therefore, a comprehensive review is needed to summarize, compare, and synthesize the factors in DPP that have been explored.

This study reviewed empirical studies published in the last decade that investigate the use of the DPP approach in various contexts, following Kitchenham

and Charters' (2007) guidelines on systematic literature review. The overarching research purpose of this review is to understand various themes and factors that play a role in the team effectiveness of DPP and, thus, to inform future research and practices on how to better leverage this approach in programming and CT education. Specifically, this study addressed four guiding questions:

RQ1. What are the research contexts of the selected empirical studies on DPP?

RQ2. How effective has DPP been when used in empirical studies?

RQ3. What themes and factors have been explored in the empirical studies on DPP?

RQ4. What effect does each factor play in the team effectiveness of DPP?

A review of existing literature

Collocated and distributed pair programming

The pair programming technique was initially used in agile software development, referring to two programmers sitting side by side and working collaboratively on the same design, algorithm, code, or test (Beck, 2000). Two programmers share one set of computer devices, including a screen, keyboard, and mouse. They typically work alternatively between two roles after a specific time period: one works as the "driver" who is responsible for typing the code and controlling the resources and devices; the other, who is called the "navigator" or "observer," work for providing possible solutions for problems, monitoring the working process, and detecting coding errors (Williams & Kessler, 2002). According to the literature, pair programming is now one of the most recurrent collaborative learning strategies in both programming industry and educational settings (Echeverría et al., 2019; Salleh et al., 2014; Williams et al., 2002).

Pair programming can advance students' programming knowledge (Umapathy & Ritzhaupt, 2017) and help them to achieve better learning performance (Zhong et al., 2016; Lye & Koh, 2014; Werner & Denning, 2009; Werner et al., 2012). It also helps learners produce high-quality codes and reduce the time to complete programming tasks than using solo programming (Jun et al., 2007). Under appropriate conditions, it is beneficial for fostering learners' confidence (Hanks et al., 2004; McDowell et al., 2002; Wei et al., 2021) and higher-ordering thinking such as computational thinking (Wei et al., 2021; Zhong et al., 2016). Additionally, students taught with pair programming reported higher satisfaction and enjoyment than those working independently (Hanks et al., 2011; McDowell et al., 2002).

The prevalent use of collocated pair programming fostered the emergence of the concept of distributed pair programming. Two programmers, most probably in separate geographically locations, still work on the same project using software that allows screen sharing and communication (Baheti et al., 2002). In other words, the main features of DPP are different physical surroundings of two team members and close collaborative relationship that relied on audio or textual communicating tools (Bandukda & Nasir, 2010). Thus, one obvious advantage of DPP is that this technique allows the students to virtually communicate and achieve collaborations

without space constraints (Baheti et al., 2002; Hanks et al., 2011). Regarding the effects, some studies have investigated and reported that DPP presents the same benefits as collocated pair programming to the teaching and learning of programming, such as improving productivity (Baheti et al., 2002), code quality (Baheti et al., 2002), and the student performance in the exam (Hanks, 2005). It can also increase development productivity in actual agile practices (Ho et al., 2004) and help with producing a higher quality of codes and consequently higher-quality software products (Rosen et al., 2010).

However, Williams and Stout (2008) tried to maintain good coordination in a globally distributed agile team with programmers in Dallas, Texas, and the "remote" office in Krakow, Poland. They claimed that although continuous conversations and efficient collaborations are possible to take place in distributed teams, it is still preferred to keep them together (Williams & Stout, 2008). The interaction between programmers was presumably identified as a good sign of the effectiveness of pair programming, which shows that they are on a soundtrack of using this technique and tend to solve the problem faster and more productively (Kattan et al., 2018). As the result of the geographical distribution, it is challenging for pairs to keep smooth communication through a good synchronous channel (Canfora et al., 2006), to support the coordination between the pairs (Rosen et al., 2010), and to reduce the time spent on switching between tools and improve the efficiency of DPP implementations (Canfora et al., 2006; Ho et al., 2004). Dealing with the difficulties and adopting the DPP approach effectively in educational settings still need further exploration (Kattan et al., 2018).

Theoretical foundation: social constructivism

Social constructivism broadly explains how learning occurs through interactions and collaborations in social activities and emphasizes students' intentions to develop their disciplinary and metacognitive skills (Kozulin, 2003). The concept of collaborative learning came from social constructivist theory, which is an instructional method that engages students to work together in groups or pairs to reach common academic goals, such as investigating a question, solving a problem, or designing a project (Barkley, 2005). During the collaboration process, students are responsible for their own learning and for helping others and subsequently achieving the success of their groups (Gokhale, 1995). The literature illustrates that collaborative learning is an effective pedagogy in introductory programming classrooms (DeClue, 2003; McDowell et al., 2002). The increasing complexity and rapid inherent change in the computer science discipline make this approach specifically appropriate for this subject (Preston, 2005).

Pair programming where students work for a shared goal is an approach of collaborative learning (Preston, 2005). In this process, the driver and the navigator work together to analyze problems, make proposals to solve the issues and test the solution design. They are not waiting for instructors to impart knowledge; instead, they will actively explore possible solutions and construct meaningful knowledge through peer communications, which aligns with Bansal's (2018) argument that the focus

of collaborative learning shifts from the teachers to the students. The cooperation mechanism in small groups and the relationship between the pairs were discussed as critical attributes of collaborative learning. Preston (2005) uses the key attributes of collaborative learning as a framework of analysis for reviewing previous studies for assessing and enhancing pair programming as a supportive pedagogy. And the findings showed that this pedagogy is a model of collaborative learning (Preston, 2005). Hanks et al. (2011) claimed that pair programming as a collaborative and explanatory activity could lead to deeper-level thinking and addressed the perceived impact on learning. The social aspects of pairing were also proved to help increase students' retention and success in later courses (Hanks et al., 2011).

The cognitive apprenticeship theory also arises from social constructivism (Edwards et al., 2010) and explains the process that an expert of a skill instructs an apprentice to master this skill. This theory underlines students' socialization into authentic practices through hands-on activity and social interactions (Collins et al., 1988). As Edwards et al. (2010) put it, "there are powerful affinities in the learning strategies behind both cognitive apprenticeship theory and pair programming" (p. 50). The approach of cognitive apprenticeship advocates that learning factual concepts through communication in situated learning helps students to understand the concepts deeply and know how to solve similar problems in the real world (Collins et al., 1988). In the same way, most programming tasks are based on realistic and complex problems that need to be solved by paired programmers in both collocated and distributed academic settings.

Related work

Table 1 compared this study with related work, including two meta-analyses and three systematic literature reviews on pair programming. The number of primary studies retrieved in these related studies varied from 18 to 74, as different data sources and the search strategies were used. The meta-analysis conducted by Hannay et al. (2009) retrieved 18 primary studies published from 1998 to 2007. The systematic reviews by Salleh et al. (2010) and Da Silva Estácio and Prikladnicki (2015) included articles published respectively from 1999 to 2007 and from 2001 to

Table 1 Comparative analysis of this review and related work

Author(s)	Type	Year	Period	No. of primary studies/No. of data sources
Brereton et al.	SLR	2009	–	28/4
Hannay et al.	MA	2009	1998–2007	18/4
Salleh et al.	SLR	2010	1999–2007	73/12
Da Silva Estácio & Prikladnicki	SLR	2015	2001–2013	34/8
Umapathy and Ritzhaupt	MA	2017	2000–2014	18/5
This study	SLR	2020	After 2010	29/8

LR Literature review, *SLR* Systematic literature review, *MA* Meta-analysis

2013. Umapathy and Ritzhaupt's (2017) meta-analysis covers more updated studies published before 2014. Considering the significant development of DPP in recent years, it is necessary to systematically aggregate and analyze more empirical studies published in the last decade.

Hannay et al. (2009) focused on the effectiveness of pair programming in regard to code quality, programming duration, and effort investment. This review proved that pair programming is faster on average than solo programming in less complex tasks and yields higher quality codes when task complexity increases (Hannay et al., 2009). In the same way, Umapathy and Ritzhaupt (2017) suggested that pair programming has a positive influence on programming assignments, exams, and passing rates but didn't show results in favor of effective measures. Brereton et al. (2009) explored the significant effect of pair programming in both pass rates and retention rates, students' confidence, and enjoyment within the master's degree introductory programming project constraints. However, these three studies did not investigate deeper into under what conditions this approach could be more effectively designed and implemented.

Salleh et al. (2010) discussed pair programming as a pedagogical tool specially at the higher education level. Besides comparing pair programming and solo programming, Salleh et al. (2010) also systematically reviewed common measures used in empirical studies at a fine granularity and identified 14 compatibility factors affecting its effectiveness, including students' skill level, personality, and self-esteem.

The systematic literature review by Da Silva Estácio and Prikladnicki (2015) is the only one that focused on pair programming for geographically distributed teams. They argued that specific infrastructures are needed to mitigate the impact of the distribution and listed essential requirements that a DPP tool should have by analyzing the tools used in the empirical studies (Da Silva Estácio & Prikladnicki, 2015). This review suggested an expanded discussion on the use of technologies in DPP and indicated the need of practical guidelines for using DPP as a teaching and learning strategy (Da Silva Estácio & Prikladnicki, 2015).

Team effectiveness model

The team effectiveness model was initially adopted from the team leadership model developed by Hughes et al. (1993) in organizational settings. Hughes et al. (1993) outlined three aspects to understand team effectiveness: (1) the inputs to the team, including the characteristics of individual team members, team design, and various organizational contexts in which the teams operate, (2) the processes that teams deal with the shared tasks, and (3) the outcomes of the teamwork, which could be an artifact produced by the team or the team harmony and development. Hughes et al.'s model was renamed the team effectiveness model by Grzeda et al. (2008) and used to assess a team-building exercise in an online undergraduate course assignment. Faja (2011) then further improved the model by establishing five constructs identified from empirical evidence. This made it an appropriate conceptual framework to analyze the viability of pair programming as a team learning method.

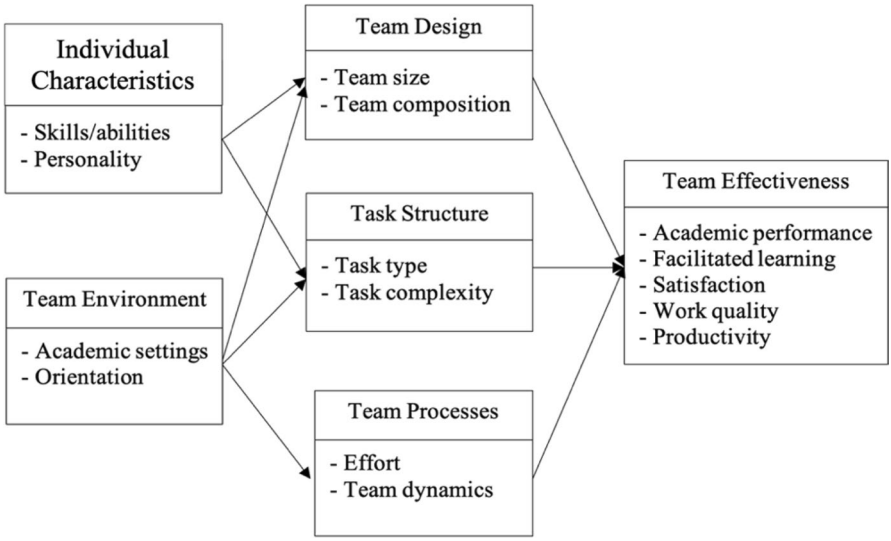

Fig. 1 Team effectiveness model in pair programming

Faja's (2011) study refined the team effectiveness model, as shown in Fig. 1. The new model argues that individual characteristics and team environment are two most fundamental components that determine the team design, team process, and task structure, and hence affect the team effectiveness in pair programming activity. This model supported us in categorizing factors systematically and answering the proposed research questions.

Individual characteristics are directly related to learners, including their prior skill and competence level (Van Toll et al., 2007), personality (Choi et al., 2009; Salleh et al., 2010), gender (Werner et al., 2004), etc. More factors under this theme will be identified in the data analysis section. Team environment consists of the academic settings and orientation of pair programming. Academic settings refer to whether the application of this approach takes place in the traditional classroom or distance education (Faja, 2011). This study focuses on the use of DPP in the distance learning setting. Faja (2011) identified the orientation approach as another factor under team environment, which can be understood as how much information about pair programming is provided by the instructor before students engage in the learning activity. It could be a short description in the syllabus (Howard, 2006), an article on pair programming (Werner et al., 2004), or an online how-to-video (Campe et al., 2019). Both the supervised training in programming task requirements before students start to work in pairs (Williams et al., 2008) and the instructor's facilitation during the pair programming activity (Howard, 2007) are critical for successful implementation of pair programming.

Team design includes the team size, which is usually two in pair programming, and team composition, which indicates how the pairs are formed (Faja, 2011). There are a variety of pair formation methods in practice. Students can be grouped into pairs either in a random manner or according to learners' characteristics and team

environment. For example, researchers and instructors can form pairs based on their common levels of programming experience to achieve better learning outcomes: Some paired students with similar competencies based on the grades (Choi et al., 2009), while some intentionally matched a strong student with a weaker one (Chigona & Pollock, 2008). They can also allow students to choose their own pairs under their guidance, with respect to their friendship and enjoyment during the programming process (Campe et al., 2019).

Faja (2011) also mentioned the task structure dimension, which mainly focuses on two points. The first is the type of the pair programming task, as pair programming has been implemented for both in and outside class activities (Howard, 2007; Simon & Hanks, 2008; Williams & Kessler, 2001). The second is the complexity of the task, even though some studies suggested that no significant impact was found when task complexity increases (Balijepally et al., 2009). From an instructional designer's perspective, learners' characteristics and team environment often determine the structure of the programming task as well (Faja, 2011).

Regarding the team processes, the effort invested by each team member is a necessary element for team effectiveness. Balijepally et al. (2009) mentioned the concept of social loafing, describing the phenomenon that students exert less effort in team work often because they assume their partner will take the responsibility and their own effort is dispensable. The team members' full engagement is required for a successful pair programming activity (Roschelle & Teasley, 1995). This is closely related to instructors' orientation and guidelines under the team environment theme (Faja, 2011). Another critical element under the team processes dimension is team dynamics, or the chemistry between two members in a team (Faja, 2011). Team dynamics is usually reflected by learners' attitudes toward collaboration and communication. Williams et al. (2008) defined two positive processes under team dynamics: pair pressure and pair relaying. The former process means team members are motivated by their partner to pay attention and keep contributing to the teamwork; while pair relaying means that working with a partner helps them better understand what they are doing or learning, since knowledge building and transfer occurs in the social interactions (Williams et al., 2008).

All these five dimensions in the model could potentially impact team effectiveness. Faja (2011) listed several elements that clearly emerged from their review, including academic performance, facilitated learning, satisfaction, work quality, and productivity. As we review the team effectiveness of the DPP approach in this analysis, more elements could be added to the model.

Methods

Systematic literature reviews are the primary approach of synthesis with the purpose not only to aggregate the existing scientific evidence on a research question but also to support the development of evidence-based practices (Kitchenham et al., 2009). Compared with a literature review, a systematic literature review is clearly defined, answers explicit research questions, and often follows a detailed and rigorous review method (Kitchenham et al., 2009; Kysh, 2013).

First, the reviews were completed individually by each author having the four research questions in mind. After that, the authors met virtually to identify any levels of dis/agreement about their analysis process and outcomes. The two authors had to agree on which studies to include or leave out of the analysis. Consensus-reaching was preferred over voting because the authors were committed to making decisions that everyone actively supported or could live with.

Search strategy and selection criteria

In this systematic review, the primary data were collected by searching for related articles published in the last decade across eight databases (ACM Digital Library; IEEE Xplore; ISI Web of Science; Science Direct; ERIC; Education Research Complete; Academic Search Complete; and Education Full Text). The search string was mainly constructed by major sets of keywords in our research questions, which were connected by Boolean AND. The synonyms or alternative terms mentioned in previous related work (e.g., Kitchenham et al., 2009; Mendes, 2005) were added and incorporated by Boolean OR.

Keywords 1 "Virtual Pair Programming" OR "Remote Pair Programming" OR "Distributed Pair Programming".

Keywords 2 experiment OR measurement OR evaluation OR assessment.

Keywords 3 effective OR efficient OR successful.

Keywords 4 empirical research OR empirical study OR data OR sample OR participants.

Experimenting with several different search criteria using various combinations of strings to obtain a manageable number of relevant literatures, we carried out an exhaustive primary search with different search strings. The complete search strings and the returned articles in each of the databases are shown in Table 2.

We accomplished the primary article selection process in two stages. In the first stage, we reviewed the title and abstract of each article to determine their suitability. As a result, publications that were irrelevant to our research questions were excluded. In the next stage, we exposed publications included during the first stage

Table 2 Digital libraries, search strings, and search results

Database	Search string	No. of studies returned
ACM digital library	(Keywords 1) AND (Keywords 2) AND (Keywords 3) AND (Keywords 4)	37
IEEE xplore	(Keywords 1) AND (Keywords 2)	291
Web of science	Keywords 1	28
Science direct	Keywords 1	10
Education research complete	*TX* Keywords 1	15
Academic search complete	*TX* Keywords 1	9
Education full text	*TX* Keywords 1	3
Eric	*TX* Keywords 1	4

to a more thorough reading of the conclusions, and in some cases, the full text. Upon completion of the primary selection, the identification of relevant literature continued with the secondary search, where all the selected articles and references in the articles identified from the primary sources were reviewed. If an article was found to be suitable, it was added to the existing list of studies qualified for the synthesis.

We used a test–retest approach in the article selection process, which means each researcher conducted the selection twice at different times to improve reliability. All the retrieved articles from the primary and secondary search were reviewed based on the exclusion criteria: (1) Articles in other types other than peer-reviewed articles; (2) Articles in languages other than English; (3) Articles without available full texts; (4) Articles published before 2010; (5) Articles with no supporting empirical evidence on DPP effectiveness or alternative terms.

Quality assessment and data extraction

To facilitate our data extraction process, we followed the quality checklist in Table 3. The criteria are composed of seven measurements adapted from Dybå and Dingsøyr's systematic review (2008) and Da Silva Estácio and Prikladnicki (2015). Instead of continuing to use the original dichotomous ("yes" or "no") scale, each of the seven items in this checklist was graded on the following scale adapted from Salleh et al.'s systematic literature review (2010): "yes" is equal to 1 point, "partially" is weighted 0.5, and "no" means 0 points. According to the checklist, the total result of each study ranged from 0 to 7, indicating the quality varied from very bad to very good.

We used a data extraction spreadsheet developed in Google Sheets to extract, record, and manage the data collected. Categories included article title, publication year, author(s), source/type (journal, conference, workshop), research purpose, research methodology, number of participants, researcher context, research

Table 3 Checklist for quality assessment of the studies

Item	Scale
1. The article is properly referenced (presents work related and literature review)	Yes = 1 partially = 0.5 no = 0
2. The objective of the research is clear	Yes = 1 partially = 0.5 no = 0
3. The research method was appropriate for achieving the objectives of the research	Yes = 1 partially = 0.5 no = 0
4. There is a clear description of the context in which the research was conducted	Yes = 1 partially = 0.5 no = 0
5. Data collection was done properly	Yes = 1 partially = 0.5 no = 0
6. The data analysis was performed properly	Yes = 1 partially = 0.5 no = 0
7. The results have credibility	Yes = 1 partially = 0.5 no = 0

The adapted checklist to assess the articles used in this systematic literature review (Da Silva Estácio et al., 2015, p. 4)

contribution, tools/infrastructures, status (included or excluded) and corresponding justifications.

Results and discussion

The following paragraphs summarize this study's results and analyze them against existing research. This section is organized around quality assessment results and results per research question, including a deep dive into the Team Effectiveness model.

Quality assessment results

The initial search of relevant literature with keyword strings produced 397 studies. Of these studies, only 45 were potentially relevant and valuable based on screening the title and abstract and removing duplicate studies. Then we read in full all the 45 articles and applied the inclusion and exclusion criteria. We selected 23 studies for the quality assessment out of 45; therefore, 23 studies were included for the synthesis of evidence. Table 4 shows the quality scores for all 23 studies. Eleven studies (48%) were considered high quality (above average), and nine studies (39%) were deemed very good and good quality, respectively; three studies (13%) attained acceptable quality.

RQ1: What are the research contexts of the selected empirical studies on DPP?

The results show that most studies (n = 16) conducted experiments with controllable variables. Four of them were case studies, and the remaining three used surveys to achieve their research objectives (see Table 5). From the distribution of the studies over the years from 2010 to 2020 in Fig. 2, we can see that compared with the first half of the last decade, there was an increased average number of empirical studies on DPP in the second half. At least two related articles were published each year after 2015, and they reached the peak (n = 4) of the number of articles respectively in 2015 and 2019. Among these studies, more than half were shared at conferences (n = 12) and workshops (n = 1) in the field of computer science and engineering education; The rest of the studies were published in journals (n = 9) and magazines (n = 1).

Table 4 Quality assessment scores

Quality scale	Very poor (< 2)	Poor (2–< 3)	Acceptable (3–< 5)	Good (5–< = 6)	Very good (> 6)	Total
No. of studies	0	0	3	9	11	23
Percentage (%)	–	–	13%	39%	48%	100%

Table 5 Distribution of studies by research design and participants

Research design	No. of studies
Experiment	16
Case study	4
Survey	3
Participants	**No. of studies**
Undergraduate	17
Graduate	2
Programmer	2
Undergraduate & graduate	1
Undergraduate & programmer	1

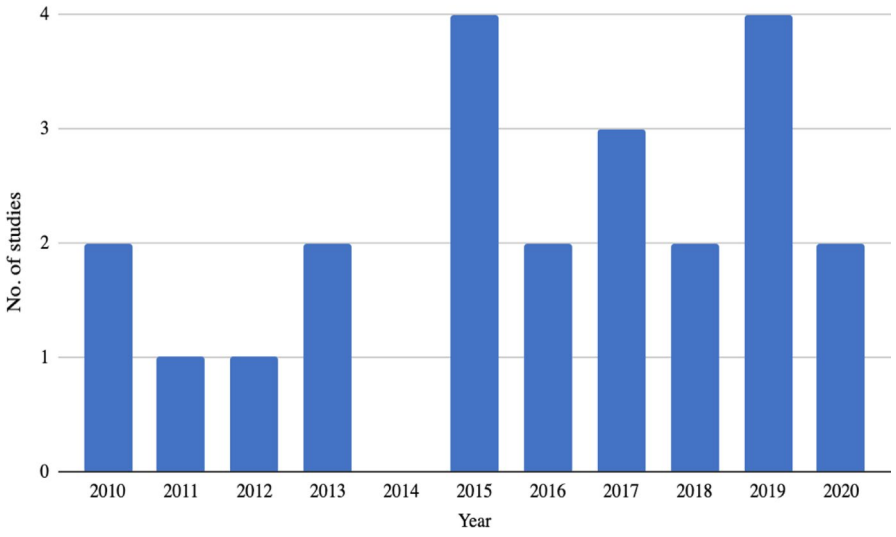

Fig. 2 Distribution of studies by years

Most of the studies were conducted at the higher education level: 17 studies recruited undergraduate students as the research participants; two studies focused on the graduate student group; and two studies recruited both undergraduate and graduate participants (see Table 5). By further looking into the research settings, we found that 13 studies were conducted in computer science courses, such as Introduction to Computer Programming, Object-Oriented Programming, Data Science, and Data Visualization. The rest of the studies focused on professional programmers in the industry: Rajpal (2018) studied a case where DPP was extremely effective for professional programmers; Dominic et al. (2020) recruited software engineers to participant in their comparison experiments and found they performed more efficiently using the virtual reality system in remote collaboration; And Bravo et al. (2013) designed collaborative programming activities with

the groupware tool called COLLECE for university students of computer science major as well as employees of a software factory, making sure that the tool is applicable to both educational and industrial settings. However, as it is advocated to equip children with programming skills and CT abilities at an early age (Kafai & Burke, 2014); more empirical research in K-12 education is in need.

Regarding the programming languages used in research intervention, excluding three articles that did not indicate which language they used, most of the studies explored DPP as a technique to program with Java (n = 16). It is not only because Java has been one of the prevalent programming languages over the last decade, but also many current integrated development environments (IDEs) were written in Java. Besides, there are respectively one study using SQL, Python, R, and a visual programming language (Alice).

RQ2: How effective has DPP been when used in empirical studies?

Of the 23 studies selected, seven focused on the impact of the DPP by comparing this approach with control groups in collocated pair programming or individual programming settings or by observing different cases where DPP was conducted. DPP, under favorable conditions, is beneficial for students to produce code of higher quality (e.g., Zacharis, 2010) and show higher levels of enjoyment during the learning experience (e.g., Xinogalos et al., 2017). However, Zacharis (2010) indicated that 57% more effort was needed from pairs to complete the same programming task than individuals. Tsai et al. (2015) suggested that students' germane cognitive load cannot be significantly reduced by sharing the workload. As a result of the complex nature of DPP, to fulfill its advantages as a learning approach and mitigate the disadvantages, suitable interventions and facilitation provided by the instructor when implementing this approach are critical (Sun et al., 2020).

RQ3: What themes and factors have been explored in the empirical studies on DPP? and RQ4: What effect does each factor play in the team effectiveness of DPP?

The remaining studies explored 12 factors that possibly have a bearing on the effectiveness of DPP. These factors were divided into four themes in the Team Effectiveness model: individual characteristics (8 factors), team design (1 factors), team process (1 factors), and team environment (2 factors). The factors investigated in these studies, as well as the study ID, were classified based on the, as shown in Table 6 and Appendix. Each study investigated the effects (e.g., significant effect, no significant effect) of several factors on several different team effectiveness measurements (e.g., academic performance, code quality, completion time). The specific effects of each factor on team effectiveness are shown in Table 7. In addition, Fig. 3 offers a visualization of team environment factors and effects.

Table 6 Investigated factors and the corresponding number of studies

Factors	Total studies	Significant effect	No significant effect
Individual characteristics			
Programming experience	2	S13	S13, S7
Actual skill	5	S4, S12, S17, S19, S21	–
Perceived skill	3	S4, S12, S13	–
Gender	2	S4, S8	–
Confidence	2	S13, S17	–
Self-esteem	1	S12	–
Time management	1	S4	–
Personality	1	S4	–
Team design			
Compatibility	2	S16, S21	–
Team process			
Feel-good	2	S13, S17	–
Team environment			
Orientation	5	S9, S15, S16, S18, S19	S15
Technology	6	S1, S2, S3, S6, S7	S14

S1 means Study [1] (Al-Jarrah and Pontelli, 2014) in the Appendix, S2 means Study [2] (Bravo et al., 2013) in the Appendix, and so on

Theme 1: individual characteristics

Aligned with Hanks et al.'s (2011) findings in the pair programming area, this analysis indicates that factors related to participants' individual characteristics attracted major investigations in the selected articles, including participant's prior programming experience, actual skill level, skill level perceived by their partner, gender, feel-good factor, confidence level, team management competency, self-esteem, and personality.

Programming experience Satratzemi et al. (2018) revealed a significantly positive impact of students' programming experience on reducing the time of solving DPP problems. Howver, according to the results of these studies, the prior programming experience has no impact on their "feel-good" factor (Tsompanoudi et al., 2019; Satratzemi et al., 2018), which is a concept represents how comfortably team members feel during the process of DPP (Muller & Padberg, 2004).

Actual skill level Based on the argument that students' actual skill level of programming is a determinant factor in pair programming, researchers hypothesized and confirmed that students' actual skill, typically measured by their scores in previous programming courses, has a positive relationship with their overall academic performance in a DPP activity (Xinogalos et al., 2019). High actual skills also contributed to the compatibility of distributed pairs (Satratzemi et al., 2019; Dou and He, 2010). Urai et al. (2015) explored the influence of the actual skill level by comparing cases

Table 7 Team environment factors and the corresponding effects

Factors	Academic performance	Program quality	Completion time	Satisfaction	Learning experience	Effort distribution	Interaction quality	Engagement	Participation rate	Knowledge acquisition	Confidence	Compatibility	Feel-good
Individual characteristics													
Programming experience			Y										N
Actual skill	Y			Y								Y	Y
Perceived skill				Y								Y	
Gender						Y						Y	
Personality												Y	
Time management												Y	
Confidence	Y		Y										
Self-esteem												N	
Team design													
Compatibility	N			Y									
Team process													
Feel-good factor	N		N								N		

Table 7 (continued)

Factors	Academic performance	Program quality	Completion time	Satisfaction	Learning experience	Effort distribution	Interaction quality	Engagement	Participation rate	Knowledge acquisition	Confidence	Compatibility	Feel-good
Team environment													
Orientation	M	Y	M			Y		Y	N	Y			
Technology	N	Y	Y	Y	Y		Y						

"Y(Yes)" means that the factor had a significant effect in one or more reviewed articles; "N(No)" means that no significant effect was shown in the reviewed articles; "M(Mixed)" means some of the articles showed significant effects, but some showed no significant effect

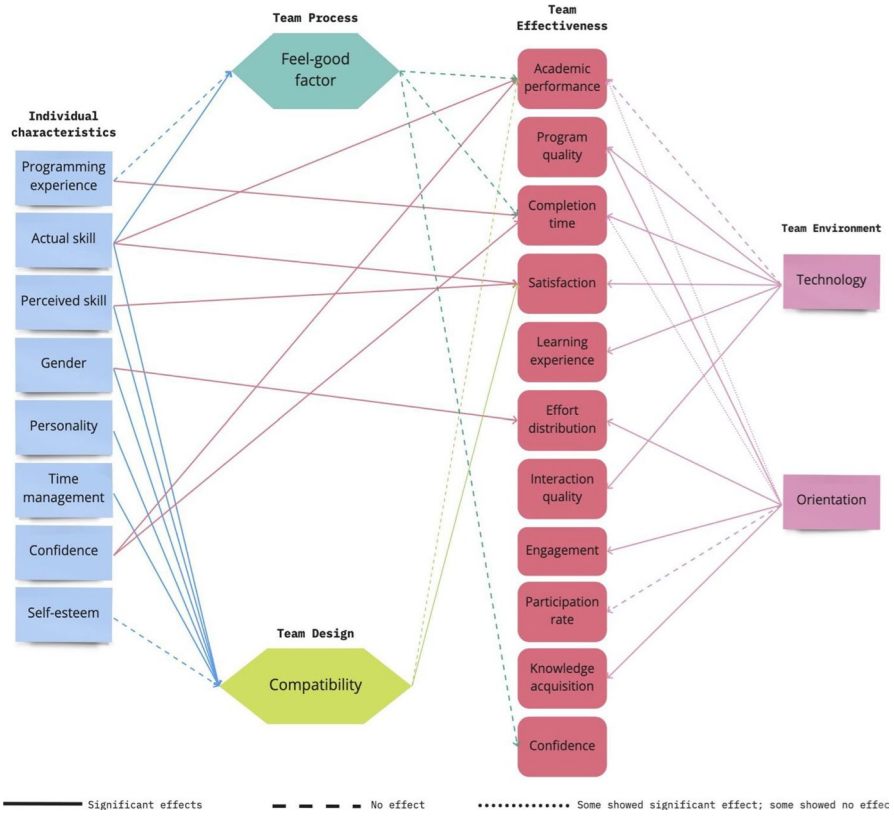

Fig. 3 Visualization of team environment factors and effects

where pairs' abilities were different with those where pairs have nearly equal ability. Qualitative questionnaires used in their study caught fine-grained self-reported data from students and provided more evidence for the complex impact of actual skill in DPP (Urai et al., 2015). Although DPP was more effective for students to learn to code whose abilities were minimally different, there are still some equal-ability pairs who reported dissatisfaction, difficulty, or worries about communicating with their team members (Urai et al., 2015). Also, in different-ability cases, the students with lower abilities felt discouraged and gradually lost motivation for the DPP session. In comparison, the stronger student had no frustration whenever playing the driver or the navigator role (Urai et al., 2015). We interpret that as the actual skill level had a significant positive impact on the feel-good factor. Additionally, Tsompanoudi et al. (2016) mentioned that groups formed based on students' similar grades in programming courses tend to feel more satisfied with the collaboration experience.

Perceived skill level Aligned with Katira et al.'s findings (2005) in collocated pair programming, Satratzemi et al. (2019) suggested that the perceived technical competence by the partner is also a significant contributor to the compatibility of team

members in DPP. Similarly, Dou and He (2010) indicated that students would be more likely to pair with peers who they thought have similar or higher skill levels, in which cases the team compatibility was higher. Though, this situation would cause frustrations if both students in a pair perceived similarly low skills (Dou & He, 2010). This factor is also positively associated with students' satisfaction toward the learning activity, as Satratzemi et al. (2018) argued that pairs who considered themselves to have comparable technical competence tend to show more satisfaction with DPP sessions.

Gender Gender was also identified as a significant individual characteristic that impacted the compatibility of pairs (Dou & He, 2010). Kuttal et al. (2019) focused closely on the influence of gender in technology mediated DPP and revealed more interesting findings: Females tended to use more non-verbal cues for communication than males and preferred to work in same-gender groups (Kuttal et al., 2019). Gender also influences the effort investment and hence determines the social dynamics within a team. Kuttal et al. (2019) compared same-gender pairs with mixed-gender pairs and found that same-gender teams tended to be democratic and equitably divide tasks; In contrast, in mixed-gender pairs, an authoritative member normally plays a leading role (Kuttal et al., 2019).

Confidence and self-esteem Physically distributed teams could be highly compatible and successful, but it is necessary to consider programmers' confidence and self-esteem as preconditions. Pairs with a higher confidence level in programming tend to achieve better performance in exams and assignments (Satratzemi et al., 2018; Tsompanoudi et al., 2019). Team members who have more confidence are likely to lead the work and reduce the implementation time of tasks (Satratzemi et al., 2018; Tsompanoudi et al., 2019).

Self-esteem and confidence are two overlapped but different concepts, as confidence focuses on learners' belief in their abilities and self-esteem is about whether learners value themselves. Satratzemi et al.'s (2019) explored whether self-esteem is a potential influencing factor for pair compatibility but found that compatibility is not dominated by students' self-esteem in remote collaboration (2019). A possible explanation for this finding is that people's self-esteem might increase as a result of rich and high-quality social interactions with others in a highly compatible team (Denissen et al., 2008).

Time management plan and personality type Because learners spend limited time learning basic concepts and principles in the DPP classroom and collaborate for a significant amount of time after the lecture, Dou and He (2010) suggested that learners prefer to pair with others who have similar time management plans outside the classroom. In addition, they also mentioned personality type as a necessary consideration to form highly compatible and successful pairs (Dou & He, 2010). They argued that a DPP tool should be able to support the evaluation of pair compatibility by analyzing factors including personality type (Dou & He, 2010); However, the authors did not demonstrate the instrument they used to measure

personality type or the rules that indicate which personality types could be compatible with each other. To better leverage this factor in future DPP practices, more research and empirical evidence are needed.

Theme 2: team design

Compatibility Investigations have also been found among the reviewed articles on how team design, particularly the pair compatibility factor, impacts the team effectiveness of DPP. Pair compatibility showed significant impact on satisfaction (Tsompanoudi et al., 2016). While Xinogalos et al. (2019) claimed that pair compatibility has no association with students' academic performances, researchers have been interested in finding patterns to improve the likelihood of forming compatible pairs in pair programming. For example, Williams et al. (2006) recruited 1350 undergraduate and graduate students and examined their compatibility. The articles reviewed in this study also discussed several relevant individual factors. However, it is still a challenge to foresee and avoid or correct a potential incompatibility between team members, no matter in collocated or distributed pair programming contexts (Dou & He, 2010).

Theme 3: team process

Feel-good factor We have talked about the individual characteristics that potentially lead to a better subjective feeling towards the distributed teamwork process. As a factor under the team process theme, the feel-good factor does not affect programmers' performance and completion time in DPP (Tsompanoudi et al., 2019; Satratzemi et al., 2018). Neither Tsompanoudi et al. (2019) nor Satratzemi et al. (2018) found any relationship between the feel-good factor reported in students' evaluation of the DPP sessions and their self-confidence in programming after the learning activity. Nevertheless, it is still a topic worthy of attention and research because it is closely related to the learners' or programmers' motivation in DPP (Chong & Ahmed, 2015).

Theme 4: team environment

Regarding theme environment and the impact of environment factors, orientation and technology were found as the main factors impacting team effectiveness of DPP.

Orientation The influence of team environment factors, including the instructional orientation and technology use in DPP, also spurred a multitude of research studies. Scripted roles orientation, meaning that students switch their roles between "driver" and "navigator" after each task for equal engagement in relevant roles, has been well-investigated in the selected articles. Consistent with research findings in the collocated pair programming area, the scripted collaborative learning approach was found quite useful in DPP contexts in reducing the average time student spent

typing program code (Tsompanoudi et al., 2013) and improving their group's performance (Tsompanoudi et al., 2013; Urai et al., 2015).

The Eclipse-SCEPPSys system used collaboration scripts as an orientation strategy in DPP, through which students in the experimental group switched their roles more often, showed higher engagement, and reached more balanced task distribution and knowledge acquisition (Tsompanoudi et al., 2015). The role switches mechanism in structured collaboration did not take more time in completing the assignments as expected, but it did not significantly shorten the completion time of assignments either (Tsompanoudi et al., 2015). However, the structured collaboration did not necessarily improve students' participation rate, and the experimental and control groups achieved similar learning performances (Tsompanoudi et al., 2015). Although students gave some positive feedback, many complained about unfair role assignment in the system because of the varying difficulty levels of tasks (Tsompanoudi et al., 2015). Tsompanoudi et al. (2015) and then improved their system accordingly.

After evaluating and validating the SCEPPSys system in their previous study, Tsompanoudi et al. (2016) then compared the efficiency of two different task distribution strategies. One approach is the scripted role policy, the other is an adaptive distribution of tasks where students decide their role depending on whether their skills match the task type. The spontaneous role assignments turned out to be more helpful in improving pass rates and learning performance in the final grades (Tsompanoudi et al., 2016). While ensuring frequent role switches and equal contributions from each student, the adaptive distribution can expose students to various learning objectives and maximize their potential compared to the scripted roles (Tsompanoudi et al., 2016). Therefore, the same as the proposition in the general area of computer-supported collaborative learning, scripted roles should be gradually fade out as programmers get more familiar with this learning mode and become growingly cooperative (Kobbe et al., 2007, as cited in Tsompanoudi et al., 2013).

Finally, automatic interventions were provided to students, and this approach significantly improved the program quality and shortened the completion time (Qiao & Bai, 2017). Computers implemented these interventions automatically based on learners' behaviors to prevent unexpected behaviors that potentially impact the programming outcomes (Qiao & Bai, 2017). Qiao and Bai (2017) indicated the difficulty of putting the new behavioral perception and automatic intervention technique into DPP practices due to the ambiguous and arbitrary content in the computer science subject, the existence of interference factors, and the complex nature of collaborative programming behaviors. Still, this study addressed the importance of orientations from the external environment.

Technology Table 8 shows that Eclipse is the mainstream IDE to support the application of DPP activities. Of the 23 articles, 13 mentioned the use of Eclipse or its freely available plugins. SCEPPSys is the most popular plugin, and it was developed as a particular DPP system that combines collaboration scripts and supports different ways of remote collaboration (Tsompanoudi et al., 2015). It also has enhanced logging functions, which is a critical function for collecting and evaluating learners' interactions within educational contexts (Tsompanoudi et al., 2015). Saros (Urai

Table 8 Distribution of studies by technologies

	Type	No. of studies
Eclipse-SCEPPSys	IDE plugin	8
Eclipse-Saros	IDE plugin	1
Eclipse-XPairtise	IDE plugin	1
Eclipse-XecliP	IDE plugin	1
Eclipse	IDE	2
Jimbo	IDE	1
COLLECE	IDE	1
AliCe-VilLagE	IDE plugin	1
TeamViewer	Remote desktop sharing	1
Skype	Telecommunications	1
NetMeeting	Telecommunications	1
Connect now	Telecommunications	1
StarLogo TNG	Modeling and simulation	1
VirtualDesk	Virtual reality environment	1

"IDE" means Integrated Development Environment

et al., 2015), Xpairtise (Qiao & Bai, 2017), and XecliP (Tsompanoudi et al., 2013) were also mentioned respectively by one article and proved to be useful plugins in DPP activities. Besides Eclipse, some studies applied other IDEs, including COLLECE, Jimbo, and an extension of Alice named Alice-VilLagE.

Three articles focused on investigating how IDEs support DPP and reported its positive impact on students' code quality (Bravo et al., 2013), completion time (Al-Jarrah & Pontelli, 2016), satisfaction (Ghorashi & Jensen, 2017; Al-Jarrah & Pontelli, 2016), and learning experience (Ghorashi & Jensen, 2017). The most significant benefit of IDEs that were specifically designed for DPP is to mitigate barriers to communication caused by the physical separation and thus improve the productivity for completing higher-quality programs. Although programmers may take more time to complete their tasks due to increased communications and coordinations, the additional time could be offset by fewer coding time (Bravo et al., 2013).

A free-text chat or an alternative audio tool is normally embedded in the above-mentioned IDEs. When there is not an IDE specially designed for DPP, it sometimes requires incorporating a tool to support communication and collaboration (Rajpal, 2018). Tools that enable teleconferencing, instant messaging, remote desktop sharing and controlling were frequently used, such as Skype (Urai et al., 2015), NetMeeting (Zacharis, 2010), Connect Now (Edwards et al., 2010), and TeamViewer (Kuttal et al., 2019).

Tsai et al. (2015) used an extension of the Logo programming language named StarLogo TNG within DPP. StarLogo TNG can model and simulate a 3D world. Even though there was no significant difference on learning performance between various programming modes shown in Tsai et al.'s (2015) study, they claimed the importance of optimizing learning support and knowledge sharing tools for DPP. Likewise, Dominic et al. (2020) brought up a novel idea of gathering learners from different physical locations in a shared virtual reality environment called

VirtualDesk. With VirtualDesk, students solved almost twice as many bugs as those who only used a screen sharing tool in a shorter period (Dominic et al., 2020).

Moreover, Bravo et al. (2013) addressed the importance of a series of awareness support mechanisms provided by technologies in DPP scenarios, among which sharing selection is vital. It provokes a visual response from the non-selector, establishes a common understanding of referring to the selected objects, and eventually promotes interaction quality (Jermann & Nüssli, 2012). Visual representations particularly benefit novice programmers when learning coding, for example, the comparison experiments by Kaplan and An (2005) showed that students with visual models produced better coding and debugging abilities than factual models. A visual execution notation in coding even strengthened beginners' competence in evaluating and solving recursive programming problems (Tung & Chang, 2001), which are the foundations for developing their CT skills.

Conclusion

We conducted a systematic review of the literature on DPP published after 2010 to identify and analyze the following:

a. Research contexts where DPP was investigated
b. Effectiveness of the DPP approach
c. Themes and factors that influence the effectiveness of DPP
d. Effects of these themes and factors.

Twenty-three articles were carefully selected, reviewed, and synthesized in this analysis through the lens of the team effectiveness model adapted from Faja (2011).

The results addressed in RQ1 showed that most studies were conducted in higher education contexts and the programming industry. However, this review found no empirical research in K-12 educational contexts. Java is the mainstream programming language in DPP classrooms and work environments; Eclipse with the SCEPPSys plugin is the most popular IDE in DPP activities.

For RQ2, seven of the 23 selected studies revealed that DPP is a valuable approach under favorable conditions, allowing programmers to produce higher-quality code and facilitate their engagement in learning activities. Compared to individual programming, DPP requires more effort from programmers because of increased communication and coordination (Zacharis, 2010). Sharing the workload in the DPP team does not apply to mitigate students' germane cognitive load (Tsai et al., 2015).

RQ3 highlighted the themes and factors that have been investigated in existing studies. A total of 12 factors under four themes were categorized, and their effects were synthesized. The results showed that individual characteristics attracted major investigations, including prior programming experience, actual skill, perceived skill, gender, personality, time management, confidence, and self-esteem. Pair compatibility is a decisive team design factor significantly affecting programmers' satisfaction. The feel-good factor in the team process was studied, but no significant impact

was found. Under the team environment theme, different opinions on the orientation (e.g., scripted roles) and the use of technology (e.g., integrated development environment tools) were compared.

The impact of individual characteristics, including participants' prior programming experience, actual skill level, skill level perceived by their partner, gender, confidence level, team management competency, self-esteem, and personality, has been reported extensively and addressed under RQ4. Additionally, Table 7 summarizes the specific effects of each individual factor as well as team compatibility, feel-good factor and team environment expressed by orientation and technology on team effectiveness.

In sum, DPP is a promising approach for learning to program and developing CT capability. However, insufficient empirical studies have investigated the use of DPP in K-12 contexts. As a result, we argue that more research in K-12 levels is in need, especially because researchers suggested that equipping children with CT abilities at an early age will give them new perspectives when solving problems in and outside school (Kafai & Burke, 2014).

Future work should also pay more attention to the under-researched factors: the design of task structure, the creation of team environment, and team processes during the DPP process (Xu & Correia, 2021). Balijepally et al. conducted a controlled laboratory experiment and reported that no significant impact was found when task complexity increases in collocated pair programming (2009). Nevertheless, how the type and complexity of learning tasks could impact the team effectiveness of DPP is still unknown.

The role of teachers was not mentioned in the reviewed studies, but this factor plays a critical role in creating a favorable team environment. Bansal addressed teachers' role in mediating, structuring, and promoting peer interaction and discussion in collaborative learning (Bansal, 2018). Both collocated and distributed pair programming strategies are believed to be most valid when teachers take the role of facilitators and guide students to collaborate effectively. The instructor should provide specific objectives for pair programming tasks, monitor pairs as they program, and provide instructional assignments from which students can learn things (Preston, 2005). In this sense, future research should explore the effect of various coaching interventions from the instructor or equivalent roles in DPP.

In addition, because DPP requires close collaborations and social interactions, the team process deserves further explorations, such that suitable interventions can be provided to achieve a healthy team ecology. For example, avoiding the social loafing that often happens to the weaker side in teamwork is always a concern (Balijepally et al., 2009; Howard, 2007). Timely detection of this problem through peer evaluations and appropriate management of pair interactions are helpful to encourage students' equal participants (Williams et al., 2008).

Significance of this study

The results of this study contribute to the body of knowledge in the literature, as it provides a snapshot of DPP research and discusses potential directions for future

research. Another key implication for educators relates to designing and implementing DPP in educational contexts. Teachers can use the results reported here to design more effective classroom interventions and capture the benefits of DPP as a collaborative programming approach to support computational thinking, a proven essential skill for the current job markets.

Considering the ever-evolving trends towards global collaboration, distance education, and teleworking in the post-peak era of the COVID-19 pandemic, programmers commonly work in a geographically distributed manner, and the demand for DPP will continue to grow, especially as computational thinking becomes a fundamental 21st-century skill. Computational thinking enables real-world problem solving and more accurate problem formulation leading to better solutions.

Appendix

Articles included in the systematic review:

[1] Al-Jarrah, A., & Pontelli, E. (2014). "AliCe-ViLlagE" Alice as a Collaborative Virtual Learning Environment. In *Proceedings of 2014 IEEE Frontiers in Education Conference (FIE)* (pp. 1–9). IEEE. https://doi.org/10.1109/fie.2014.7044089

[2] Bravo, C., Duque, R., & Gallardo, J. (2013). A groupware system to support collaborative programming: Design and experiences. *Journal of Systems and Software, 86*(7), 1759–1771. https://doi.org/10.1016/j.jss.2012.08.039

[3] Dominic, J., Tubre, B., Ritter, C., Houser, J., Smith, C., & Rodeghero, P. (2020). Remote Pair Programming in Virtual Reality. In *Proceedings of 2020 IEEE International Conference on Software Maintenance and Evolution* (ICSME) (pp. 406–417). IEEE. https://doi.org/10.1109/icsme46990.2020.00046

[4] Dou, W., & He, W. (2010). Compatibility and requirements analysis of distributed pair programming. In *Proceedings of 2010 Second International Workshop on Education Technology and Computer Science* (Vol. 1, pp. 467–470). IEEE. https://doi.org/10.1109/etcs.2010.367

[5] Edwards, R. L., Stewart, J. K., & Ferati, M. (2010). Assessing the effectiveness of distributed pair programming for an online informatics curriculum. *ACM inroads, 1*(1), 48–54. https://doi.org/10.1145/1721933.1721951

[6] Ghorashi, S., & Jensen, C. (2017). Integrating collaborative and live coding for distance education. *Computer, 50*(5), 27–35. https://doi.org/10.1109/mc.2017.131

[7] Jermann, P., & Nüssli, M. A. (2012). Effects of sharing text selections on gaze cross-recurrence and interaction quality in a pair programming task. In *Proceedings of the 2012 ACM conference on Computer Supported Cooperative Work* (pp. 1125–1134). https://doi.org/10.1145/2145204.2145371

[8] Kuttal, S. K., Gerstner, K., & Bejarano, A. (2019). Remote pair programming in online cs education: Investigating through a gender lens. In *Proceedings of*

2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC) (pp. 75–85). IEEE. https://doi.org/10.1109/vlhcc.2019.8818790

[9] Qiao, X., & Bai, X. (2017). Behavior perception and automatic intervention for cooperative learning in network environment. In *Proceedings of 2017 10th International Conference on Ubi-media Computing and Workshops (Ubi-Media)* (pp. 1–6). IEEE. https://doi.org/10.1109/umedia.2017.8074140

[10] Rajpal, M. (2018). Effective distributed pair programming. In *Proceedings of 2018 IEEE/ACM 13th International Conference on Global Software Engineering (ICGSE) (pp. 6–10)*. IEEE. https://doi.org/10.1145/3196369.3196388

[11] Saltz, J., & Heckman, R. (2020). Using structured pair activities in a distributed online breakout room. *Online Learning, 24*(1), 227–244. https://doi.org/10.24059/olj.v24i1.1632

[12] Satratzemi, M., Tsompanoudi, D., Xinogalos, S., & Karamitopoulos, L. (2019). Examining the compatibility of students in distributed pair programming. In *Proceedings of* 2019 *European Conference on e-Learning* (pp. 510-XVII). Academic Conferences. https://doi.org/10.1155/2018/6523538

[13] Satratzemi, M., Xinogalos, S., Tsompanoudi, D., & Karamitopoulos, L. (2018). Examining student performance and attitudes on distributed pair programming. *Scientific Programming, 2018*, 1–8. https://doi.org/10.1155/2018/6523538

[14] Tsai, C. Y., Yang, Y. F., & Chang, C. K. (2015). Cognitive load comparison of traditional and distributed pair programming on visual programming language. In *Proceedings of 2015 International Conference of Educational Innovation through Technology (EITT) (pp. 143–146)*. IEEE. https://doi.org/10.1109/eitt.2015.37

[15] Tsompanoudi, D., Satratzemi, M., & Xinogalos, S. (2015). Distributed pair programming using collaboration scripts: An educational system and initial results. *Informatics in Education, 14*(2), 291–314. https://doi.org/10.15388/infedu.2015.17

[16] Tsompanoudi, D., Satratzemi, M., & Xinogalos, S. (2016). Evaluating the effects of scripted distributed pair programming on student performance and participation. *IEEE Transactions on Education, 59*(1), 24–31. https://doi.org/10.1109/te.2015.2419192

[17] Tsompanoudi, D., Satratzemi, M., Xinogalos, S. & Karamitopoulos, L. (2019). An empirical study on factors related to distributed pair programming. *International Journal of Engineering Pedagogy, 9*(2), 65–81. https://doi.org/10.3991/ijep.v9i2.9947

[18] Tsompanoudi, D., Satratzemi, M., & Xinogalos, S. (2013). Exploring the effects of collaboration scripts embedded in a distributed pair programming system. In *Proceedings of the 18th ACM conference on Innovation and Technology in Computer Science Education* (pp. 225–230). https://doi.org/10.1145/2462476.2462500

[19] Urai, T., Umezawa, T., & Osawa, N. (2015). Enhancements to support functions of distributed pair programming based on action analysis. In *Proceedings of the 2015 ACM Conference onInnovation and Technology in Computer Science Education* (pp. 177–182). https://doi.org/10.1145/2729094.2742616

[20] Xinogalos, S., Satratzemi, M., Chatzigeorgiou, A., & Tsompanoudi, D. (2017). Student perceptions on the benefits and shortcomings of distributed pair programming assignments. In *Proceedings of 2017 IEEE Global Engineering Education Conference (EDUCON)* (pp. 1513–1521). IEEE. https://doi.org/10.1109/educon.2017.7943050

[21] Xinogalos, S., Satratzemi, M., Chatzigeorgiou, A., & Tsompanoudi, D. (2019). Factors affecting students' performance in distributed pair programming. *Journal of Educational Computing Research, 57*(2), 513–544. https://doi.org/10.1177/0735633117749432

[22] Xinogalos, S., Satratzemi, M., Tsompanoudi, D., & Chatzigeorgiou, A. (2016). Monitoring an OOP course through assignments in a distributed pair programming system. In *Proceedings of 2016 SQAMIA* (pp. 97–104). https://ruomo.lib.uom.gr/handle/7000/461

[23] Zacharis, N. Z. (2010). Measuring the effects of virtual pair programming in an introductory programming java course. *IEEE Transactions on Education, 54*(1), 168–170. https://doi.org/10.1109/te.2010.2048328

Declarations

Conflict of interest No potential competing interest was reported by the authors.

References

Baheti, P., Gehringer, E., & Stotts, D. (2002). Exploring the efficacy of distributed pair programming. In D. Wells & L. Williams (Eds.), *Extreme programming and agile methods—XP/Agile universe 2002 proceedings* (pp. 208–220). Springer.

Balijepally, V., Mahapatra, R., Nerur, S., & Price, K. H. (2009). Are two heads better than one for software development? The productivity paradox of pair programming. *Management Information Systems Quarterly, 33*(1), 99–118. https://doi.org/10.2307/20650280

Bandukda, M., & Nasir, Z. (2010). Efficacy of distributed pair programming. In *2010 International Conference on Information and Emerging Technologies Proceedings* (pp. 1–6). IEEE.

Bansal, S. (2018). A constructivist perspective towards collaborative learning. *International Journal of Research in Social Sciences, 8*(5), 679–689.

Barkley, E. F., Cross, K. P., & Major, C. H. (2005). *Collaborative learning techniques: A handbook for college faculty*. Jossey-Bass.

Beck, K. (2000). *Extreme programming explained: Embrace change*. Addison-Wesley Professional.

Brereton, P., Turner, M., & Kaur, R. (2009). Pair programming as a teaching tool: A student review of empirical studies. In *2009 22nd Conference on Software Engineering Education and Training Proceedings* (pp. 240–247). IEEE.

Cahapay, M. B. (2020). Rethinking education in the new normal post-COVID-19 era: A curriculum studies perspective. *Aquademia, 4*(2), ep20018. https://doi.org/10.29333/aquademia/8315

Campe, S., Green, E., & Denner, J. (2019). *K-12 pair programming toolkit*. Scotts Valley: ETR.

Canfora, G., Cimitile, A., Di Lucca, G. A., & Visaggio, C. A. (2006). How distribution affects the success of pair programming. *International Journal of Software Engineering and Knowledge Engineering, 16*(02), 293–313. https://doi.org/10.1142/s0218194006002756

Chigona, W., & Pollock, M. (2008). Pair programming for information systems students new to programming: Students' experiences and teachers' challenges. In *2008 Portland International Conference on Management of Engineering & Technology Proceedings* (pp. 1587–1594). IEEE.

Choi, K. S., Deek, F. P., & Im, I. (2009). Pair dynamics in team collaboration. *Computers in Human Behavior, 25*(4), 844–852. https://doi.org/10.1016/j.chb.2008.09.005

Chong, Y. S., & Ahmed, P. K. (2015). Student motivation and the 'feel good' factor: An empirical examination of motivational predictors of university service quality evaluation. *Studies in Higher Education, 40*(1), 158–177. https://doi.org/10.1080/03075079.2013.818643

Collins, A., Brown, J. S., & Newman, S. E. (1988). Cognitive apprenticeship: Teaching the craft of reading, writing and mathematics. *Thinking: the Journal of Philosophy for Children, 8*(1), 2–10. https://doi.org/10.5840/thinking19888129

Da Silva Estácio, B. J., & Prikladnicki, R. (2015). Distributed pair programming: A systematic literature review. *Information and Software Technology, 63*, 1–10. https://doi.org/10.1016/j.infsof.2015.02.011

DeClue, T. H. (2003). Pair programming and pair trading: Effects on learning and motivation in a CS2 course. *Journal of Computing Sciences in Colleges, 18*(5), 49–56. https://doi.org/10.5555/771832.771843

Denissen, J. J., Penke, L., Schmitt, D. P., & Van Aken, M. A. (2008). Self-esteem reactions to social interactions: Evidence for sociometer mechanisms across days, people, and nations. *Journal of Personality and Social Psychology, 95*(1), 181. https://doi.org/10.1037/0022-3514.95.1.181

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology, 50*(9–10), 833–859. https://doi.org/10.1016/j.infsof.2008.01.006

Echeverría, L., Cobos, R., & Morales, M. (2019). Improving the students computational thinking skills with collaborative learning techniques. *IEEE Revista Iberoamericana De Tecnologias Del Aprendizaje, 14*(4), 196–206. https://doi.org/10.1109/rita.2019.2952299

Edwards, R. L., Stewart, J. K., & Ferati, M. (2010). Assessing the effectiveness of distributed pair programming for an online informatics curriculum. *ACM Inroads, 1*(1), 48–54. https://doi.org/10.1145/1721933.1721951

Faja, S. (2011). Pair programming as a team based learning activity: A review of research. *Issues in Information Systems, 12*(2), 207–216. https://doi.org/10.48009/2_iis_2011_207-216

Falkner, K., Vivian, R., & Falkner, N. (2014). The Australian digital technologies curriculum: challenge and opportunity. In D. D'Souza & J. Whalley (Eds.), *Proceedings of the Sixteenth Australasian Computing Education Conference* (pp. 3–12). Australian Computer Society.

Gokhale, A. (1995). Collaborative learning enhances critical thinking. *Journal of Technology Education*. https://doi.org/10.21061/jte.v7i1.a.2

Grzeda, M., Haq, R., & LeBrasseur, R. (2008). Team building in an online organizational behavior course. *Journal of Education for Business, 83*(5), 275–282. https://doi.org/10.3200/joeb.83.5.275-282

Hanks, B. (2005). Student performance in CS1 with distributed pair programming. *ACM SIGCSE Bulletin, 37*(3), 316–320. https://doi.org/10.1145/1151954.1067532

Hanks, B., Fitzgerald, S., McCauley, R., Murphy, L., & Zander, C. (2011). Pair programming in education: A literature review. *Computer Science Education, 21*(2), 135–173. https://doi.org/10.1080/08993408.2011.579808

Hanks, B., McDowell, C., Draper, D., & Krnjajic, M. (2004). Program quality with pair programming in CS1. *ACM SIGCSE Bulletin, 36*(3), 176–180. https://doi.org/10.1145/1026487.1008043

Hannay, J. E., Dybå, T., Arisholm, E., & Sjøberg, D. I. (2009). The effectiveness of pair programming: A meta-analysis. *Information and Software Technology, 51*(7), 1110–1122. https://doi.org/10.1016/j.infsof.2009.02.001

Ho, C. W., Raha, S., Gehringer, E., & Williams, L. (2004). Sangam: A distributed pair programming plug-in for Eclipse. In *Proceedings of the 2004 OOPSLA workshop on eclipse technology eXchange* (pp. 73–77). Association for Computing Machinery. https://doi.org/10.1145/1066129.1066144

Hodges, C., Moore, S., Lockee, B., Trust, T., & Bond, A. (2020). The difference between emergency remote teaching and online learning. *EDUCAUSE Review*, *27*. https://er.educause.edu/articles/2020/3/the-difference-between-emergency-remote-teaching-and-online-learning.

Howard, E. V. (2006). Attitudes on using pair-programming. *Journal of Educational Technology Systems, 35*(1), 89–103. https://doi.org/10.2190/5K87-58W8-G07M-2811

Howard, E. V. (2007). Attitudes on using pair-programming. *Journal of Educational Technology Systems, 35*(1), 89–103. https://doi.org/10.2190/5k87-58w8-g07m-2811

Hsu, T. C., Chang, S. C., & Hung, Y. T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education, 126*, 296–310. https://doi.org/10.1016/j.compedu.2018.07.004

Hughes, R. L., Ginnett, R. C., & Curphy, G. J. (1993). *Leadership: Enhancing the lessons of experience*. Irwin.

Jun, S., Kim, S., & Lee, W. (2007). Online pair-programming for learning programming of novices. *WSEAS Transactions on Advances in Engineering Education, 9*, 187–192.

Juškevičienė, A., & Dagienė, V. (2018). Computational thinking relationship with digital competence. *Informatics in Education, 17*(2), 265–284. https://doi.org/10.15388/infedu.2018.14

Kafai, Y. B., & Burke, Q. (2014). *Connected code: Why children need to learn programming*. MIT Press.

Kaplan, D. E., & An, H. (2005). Facts, procedures, and visual models in Novices' learning of coding skills. *Journal of Computing in Higher Education, 17*(1), 43–70. https://doi.org/10.1007/BF02960226

Katira, N., Williams, L., & Osborne, J. (2005). Towards increasing the compatibility of student pair programmers. In *Proceedings of the 27th international conference on Software engineering* (pp. 625–626). Association for Computing Machinery. https://doi.org/10.1145/1062455.1062572

Kattan, H. M., Soares, F., Goldman, A., Deboni, E., & Guerra, E. (2018). Swarm or pair? Strengths and weaknesses of Pair Programming and Mob Programming. In *Proceedings of the 19th International Conference on Agile Software Development: Companion* (pp. 1–4). Association for Computing Machinery. https://doi.org/10.1145/3234152.3234169

Kitchenham, B. A., & Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering technical report. *EBSE Technical Report*. https://www.elsevier.com/__data/promis_misc/525444systematicreviewsguide.pdf

Kitchenham, B., Brereton, O. P., Budgen, D., Turner, M., Bailey, J., & Linkman, S. (2009). Systematic literature reviews in software engineering—A systematic literature review. *Information and Software Technology, 51*(1), 7–15. https://doi.org/10.1016/j.infsof.2008.09.009

Kobbe, L., Weinberger, A., Dillenbourg, P., Harrer, A., Hämäläinen, R., Häkkinen, P., & Fischer, F. (2007). Specifying computer-supported collaboration scripts. *International Journal of Computer-Supported Collaborative Learning, 2*(2), 211–224. https://doi.org/10.1007/s11412-007-9014-4

Kozulin, A. (2003). Psychological tools and mediated learning. In A. Kozulin, B. Gindis, V. Ageyev, & S. Miller (Eds.), *Vygotsky's educational theory in cultural context* (pp. 15–38). Cambridge University Press.

Kysh, Lynn (2013). Difference between a systematic review and a literature review. *figshare*. https://doi.org/10.6084/m9.figshare.766364.v1

Lin, L., Mills, L. A., & Ifenthaler, D. (2016). Collaboration, multi-tasking and problem solving performance in shared virtual spaces. *Journal of Computing in Higher Education, 28*(3), 344–357. https://doi.org/10.1007/s12528-016-9117-x

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41*, 51–61. https://doi.org/10.1016/j.chb.2014.09.012

McDowell, C., Werner, L., Bullock, H., & Fernald, J. (2002). The effects of pair-programming on performance in an introductory programming course. *ACM SIGCSE Bulletin, 34*(1), 38–42. https://doi.org/10.1145/563340.563353

Mendes, E. (2005). A systematic review of Web engineering research. In *2005 International Symposium on Empirical Software Engineering Proceedings* (pp. 498–507). IEEE. https://doi.org/10.1109/isese.2005.1541857

Muller, M. M., & Padberg, F. (2004). An empirical study about the feelgood factor in pair programming. In *10th International Symposium on Software Metrics Proceedings* (pp. 151–158). IEEE. https://doi.org/10.1109/metric.2004.1357899

Preston, D. (2005). Pair programming as a model of collaborative learning: A review of the research. *Journal of Computing Sciences in Colleges, 20*(4), 39–45. https://doi.org/10.5555/1047846.1047852

Roschelle, J., & Teasley, S. D. (1995). The construction of shared knowledge in collaborative problem solving. In *Computer supported collaborative learning* (pp. 69–97). https://doi.org/10.1007/978-3-642-85098-1_5

Rosen, E., Salinger, S., & Oezbek, C. (2010). Project kick-off with distributed pair programming. In J. Lawrance & R. Bellamy (Eds.), *Proceedings of the 22nd annual workshop of the psychology of programming interest group* (pp. 121–135). Maria Paloma Díaz Pérez and Mary Beth Rosson.

Salleh, N., Mendes, E., & Grundy, J. (2010). Empirical studies of pair programming for CS/SE teaching in higher education: A systematic literature review. *IEEE Transactions on Software Engineering, 37*(4), 509–525. https://doi.org/10.1109/tse.2010.59

Salleh, N., Mendes, E., & Grundy, J. (2014). Investigating the effects of personality traits on pair programming in a higher education setting through a family of experiments. *Empirical Software Engineering, 19*(3), 714–752. https://doi.org/10.1007/s10664-012-9238-4

Simon, B., & Hanks, B. (2008). First-Year Students' Impressions of Pair Programming. *ACM Journal on Educational Resources in Computing, 7*(4), 1–28. https://doi.org/10.1145/1316450.1316455

Sun, D., Ouyang, F., Li, Y., & Chen, H. (2020). Three contrasting pairs' collaborative programming processes in China's secondary education. *Journal of Educational Computing Research,* 59(4), 740–762. https://doi.org/10.1177/0735633120973430

Thomas, M. S., & Rogers, C. (2020). Education, the science of learning, and the COVID-19 crisis. *Prospects, 49*(1), 87–90. https://doi.org/10.1007/s11125-020-09468-z

Tsompanoudi, D., Satratzemi, M., & Xinogalos, S. (2015). Distributed pair programming using collaboration scripts: An educational system and initial results. *Informatics in Education, 14*(2), 291–314. https://doi.org/10.15388/infedu.2015.17

Tung, S. H., Chang, C. T., Wong, W. K., & Jehng, J. C. (2001). Visual representations for recursion. *International Journal of Human-Computer Studies, 54*(3), 285–300. https://doi.org/10.1006/ijhc.2000.0433

Umapathy, K., & Ritzhaupt, A. D. (2017). A meta-analysis of pair-programming in computer programming courses: Implications for educational practice. *ACM Transactions on Computing Education, 17*(4), 1–13. https://doi.org/10.1145/2996201

UNESCO, U. (2020). COVID-19 educational disruption and response. *UNESCO*.

Van Toll, T., Lee, R., & Ahlswede, T. (2007). Evaluating the usefulness of pair programming in a classroom setting. In *6th IEEE/ACIS International Conference on Computer and Information Science* (pp. 302–308). IEEE. https://doi.org/10.1109/icis.2007.96

Wei, X., Lin, L., Meng, N., Tan, W., & Kong, S. C. (2021). The effectiveness of partial pair programming on elementary school students' computational thinking skills and self-efficacy. *Computers & Education, 160*, 104023. https://doi.org/10.1016/j.compedu.2020.104023

Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012). The fairy performance assessment: Measuring computational thinking in middle school. In *Proceedings of the 43rd ACM technical symposium on computer science education* (pp. 215–220). ACM. https://doi.org/10.1145/2157136.2157200

Werner, L., & Denning, J. (2009). Pair programming in middle school: What does it look like? *Journal of Research on Technology in Education, 42*(1), 29–49. https://doi.org/10.1080/15391523.2009.10782540

Werner, L., Hanks, B., & McDowell, C. (2004). Pair-programming helps female computer science students. *ACM Journal of Educational Resources in Computing, 4*(1), 1–8. https://doi.org/10.1145/1060071.1060075

Williams, L., Layman, L., Osborne, J., & Katira, N. (2006). Examining the compatibility of student pair programmers. In *Proceedings of AGILE 2006* (pp. 420–430). IEEE. https://doi.org/10.1109/agile.2006.25

Williams, L., McCrickard, D. S., Layman, L., & Hussein, K. (2008). Eleven guidelines for implementing pair programming in the classroom. In *Proceedings of AGILE 2008* (pp. 445–452). IEEE. https://doi.org/10.1109/agile.2008.12

Williams, W., & Stout, M. (2008). Colossal, scattered, and chaotic (planning with a large, distributed team). In *Proceedings of the AGILE 2008* (pp. 356–361). IEEE. https://doi.org/10.1109/agile.2008.25

Williams, L., & Kessler, R. (2001). Experiments with industry's "pair-programming" model in the computer science classroom. *Computer Science Education, 11*(1), 7–20. https://doi.org/10.1076/csed.11.1.7.3846

Williams, L., & Kessler, R. (2002). Pair programming: Experience the difference. In D. Wells & L. Williams (Eds.), *Extreme programming and agile methods—XP/Agile universe 2002* (pp. 271–272). Springer.

Williams, L., Wiebe, E., Yang, K., Ferzli, M., & Miller, C. (2002). In support of pair programming in the introductory computer science course. *Computer Science Education, 12*(3), 197–212. https://doi.org/10.1076/csed.12.3.197.8618

Worrell, B., Brand, C., & Repenning, A. (2015). Collaboration and Computational Thinking: A classroom structure. In *Proceedings of the 2015 IEEE Symposium on Visual Languages and Human-Centric Computing* (pp. 183–187). IEEE. https://doi.org/10.1109/VLHCC.2015.7357215

Xu, F., & Correia, A.-P. (2021). A systematic review of distributed pair programming based on the team effectiveness model. In C. K. Looi, B. Wadhwa, V. Dagiené, P. Seow, Y. H. Kee, & L. K. Wu (Eds.), *Proceedings of the 5th APSCE international computational thinking and STEM in education conference* (pp. 85–86). National Institute of Education.

Zhong, B., Wang, Q., & Chen, J. (2016). The impact of social factors on pair programming in a primary school. *Computers in Human Behavior, 64*, 423–431. https://doi.org/10.1016/j.chb.2016.07.017

Fan Xu is a doctoral candidate of Learning Technologies at The Ohio State University and a Senior Learning Designer at the Center on Education and Learning for Employment (https://cete.osu.edu). Her research interests include instructional design, collaborative learning, computational thinking, learning analytics, and dyadic data analytics.

Ana-Paula Correia is a Professor of Learning Technologies and the Director of the Center on Education and Training for Employment at The Ohio State University. Specifically, her expertise in online learning, online and mobile learning, usability and evaluation, collaborative learning, and entrepreneurial educational approaches have been published in nearly 90 refereed articles and book chapters.