The Effects of Pair Programming on Middle School Students' Computational Thinking Skills and Engagement in Introductory Computer Science Classes

David M. Schieferstein

Colorado State University

EDRM 600 Introduction to Research Methods

Dr. Tobin Lopes

April 16th, 2024

Abstract

Computational thinking (CT) skills are essential for success in our technology-driven society, yet research indicates gaps in CT proficiency and engagement in computer science across demographic groups of middle school students. Pair programming, a collaborative learning approach, has shown promise as a culturally responsive pedagogical strategy to address these disparities. This mixed methods study aimed to examine the effects of weekly pair programming activities on middle school students' CT skills and engagement in introductory computer science classes. The qualitative component explored students' experiences, perceived benefits, and challenges related to pair programming through interviews and focus groups. The quantitative quasi-experimental component compared CT skills and engagement measures between classrooms assigned to weekly pair programming activities versus individual work. Results indicated that students in the pair programming group demonstrated significantly higher post-intervention CT skills and engagement compared to the individual practice group. Qualitative findings revealed that students perceived pair programming as beneficial for enhancing collaboration, peer learning, and problem-solving skills. By providing evidence for the effectiveness of pair programming in reducing demographic gaps in CT skills and engagement, this study informs best practices for inclusive and equitable computer science education in middle schools.

The Effects of Pair Programming on Middle School Students' Computational Thinking Skills and Engagement in Introductory Computer Science Classes

Computational thinking is a fundamental skill that involves solving problems and designing systems using principles from computer science. CT includes skills such as decomposing complex problems into smaller parts, recognizing patterns, abstracting essential information, and developing step-by-step solutions (algorithms). These skills are becoming increasingly important in our technology-driven society, and developing CT abilities in middle school lays a critical foundation for future success in computer science and STEM fields (Creswell & Creswell, 2022; Denner et al., 2021).

However, research indicates significant gaps in CT skills across demographic groups, with female students and students from underrepresented minority backgrounds demonstrating lower CT proficiency compared to their male and majority peers (Jain et al., 2022; Ni et al., 2023). Engaging diverse learners in computer science and equipping them with key CT competencies is crucial for promoting equity and increasing representation in computing fields.

Pair programming is a collaborative pedagogical approach where two students work together on coding tasks, alternating between "driver" and "navigator" roles (Denner et al., 2021). The "driver" writes the code while the "navigator" reviews each line, checks for errors, and considers the overall direction. This approach aligns with culturally responsive teaching practices that emphasize cooperation, communication, and shared problem-solving. Prior studies suggest pair programming can enhance CT practices like problem decomposition, pattern recognition, and algorithm design for middle school computer science students (Denner et al., 2021; Werner et al., 2013). Pair programming may also promote student engagement by

providing peer support, making coding tasks more manageable, and increasing motivation and persistence (Denner et al., 2021; Williams & Kessler, 2003).

This mixed methods study aimed to investigate how pair programming influences diverse middle school students' CT skills and engagement in introductory computer science classes compared to individual programming activities. The findings can inform instructional practices, curriculum design, and efforts to make computer science education more inclusive and equitable.

Background & Rationale

Computational thinking skills, such as problem decomposition, pattern recognition, abstraction, and algorithm design, are crucial for success in our increasingly technology-driven society (Denner et al., 2021). Developing these skills in middle school provides a foundation for future pursuits in computer science and STEM fields (Creswell & Creswell, 2022). However, research has consistently shown disparities in CT proficiency and engagement in computer science across demographic groups, with female students and students from underrepresented minority backgrounds demonstrating lower levels compared to their male and majority peers (Jain et al., 2022; Ni et al., 2023; Google Inc. & Gallup Inc., 2016).

Culturally responsive pedagogies have emerged as a promising approach to address these gaps and engage diverse learners in computer science (Madkins et al., 2020; Scott et al., 2015). Pair programming, a collaborative learning strategy where two students work together on coding tasks by alternating "driver" and "navigator" roles, aligns with culturally responsive teaching principles that emphasize cooperation, communication, and shared problem-solving (Denner et al., 2021; Werner et al., 2013). The "driver" writes the code while the "navigator" reviews each line, checks for errors, and considers the overall direction. This approach allows students to learn from each other, share knowledge, and develop CT skills collaboratively.

Several studies have explored the potential benefits of pair programming for enhancing CT skills and engagement in computer science education. Denner et al. (2021) found that pair programming helped middle school students develop CT practices like problem decomposition, pattern recognition, and algorithm design. Specifically, students who participated in pair programming activities were better able to break down complex problems into smaller, more manageable parts (decomposition), identify similarities and differences in code (pattern recognition), and create step-by-step instructions for solving problems (algorithm design) compared to those who worked individually. These findings suggest that pair programming can be an effective strategy for promoting key CT skills, which directly relates to our research question on the impact of pair programming on CT skills.

Werner et al. (2013) reported that pair programming increased student engagement and persistence in computer science projects. In their study, middle school students who worked in pairs on coding tasks demonstrated higher levels of engagement, as measured by observations of on-task behavior and student self-reports, compared to those who worked alone. Additionally, students in the pair programming condition were more likely to persist in the face of challenges and complete their projects successfully. These results align with our research question on the effects of pair programming on student engagement and highlight the potential for pair programming to promote motivation and resilience in computer science learning.

Furthermore, pair programming has been shown to promote peer support, make coding tasks more manageable, and increase motivation (Williams & Kessler, 2003; McDowell et al., 2006). For example, McDowell et al. (2006) found that university students who participated in pair programming had higher course completion rates, better performance on programming assignments, and increased confidence in their coding abilities compared to those who worked

individually. These findings, while from a higher education context, suggest that the benefits of pair programming for engagement and persistence may extend across different age groups and educational levels.

However, there is a need for more research directly examining the effects of pair programming on reducing demographic gaps in CT skills and engagement in middle school computer science settings. This mixed methods study aimed to fill this gap by investigating how pair programming influences diverse middle school students' CT skills and engagement in introductory computer science classes compared to individual programming activities. By exploring the experiences and outcomes of pair programming for different demographic groups, this study can inform the development of inclusive and equitable instructional practices in middle school computer science education.

The findings of this study can contribute to the growing body of literature on culturally responsive pedagogies in computer science education and provide practical guidance for educators seeking to promote CT skills and engagement among diverse learners. By examining the effects of pair programming on both CT skills and engagement, this study can offer a more comprehensive understanding of how this collaborative learning strategy can support student success and persistence in computer science. Ultimately, this research can inform efforts to broaden participation in computing fields and ensure that all students have access to high-quality computer science education that prepares them for future academic and career opportunities.

Methods

Study Design and Participants

The study involved 102 middle school students spread across six introductory computer science classes at Cedar Park Middle School (CPMS) in Cedar Park, Texas. Classes were

randomly assigned to either the pair programming condition or the individual practice condition, with three classes in each condition. This quasi-experimental design allowed for causal inferences about the effects of pair programming while working within the constraints of existing class structures (Creswell & Creswell, 2022).

Curriculum and Coding Activities

The introductory computer science curriculum focused on developing CT skills and programming concepts using the MIT App Inventor platform. The CT concepts covered included abstraction, decomposition, pattern recognition, and algorithm design. These concepts were introduced through a combination of direct instruction, guided examples, and hands-on coding activities.

Students learned and practiced programming skills such as variables, conditionals, loops, and functions. These skills were introduced incrementally, with each new concept building upon the previous ones. Students engaged in a variety of coding activities, ranging from simple exercises to more complex projects that required them to apply multiple programming concepts.

In the pair programming condition, students completed coding activities with an assigned partner each week. Pairs were rotated every two weeks to give students opportunities to work with different peers. Each pair programming session lasted for 30-40 minutes, with students alternating between the "driver" and "navigator" roles every 10-15 minutes. The "driver" was responsible for writing the code, while the "navigator" reviewed the code, provided feedback, and offered suggestions. Students were provided with guidelines and prompts to facilitate effective collaboration, such as discussing the problem, brainstorming solutions, and explaining their thought processes.

In the individual practice condition, students completed the same coding activities independently, without a partner. They had the opportunity to ask the teacher for assistance if needed, but were encouraged to try to solve problems on their own first.

Measures

Computational Thinking Skills

CT skills were assessed using the Computational Thinking Test (CTT) (Román-González et al., 2017). The CTT is a multiple-choice instrument that measures students' proficiency in key CT concepts, such as abstraction, decomposition, pattern recognition, and algorithm design. The test has been validated with middle school populations and has demonstrated good reliability (Cronbach's $\alpha = .82$) and convergent validity with other measures of CT (Román-González et al., 2017).

Student Engagement

Student engagement was measured using an adapted version of the Computer Science Engagement Survey (CSES) (Wiebe et al., 2018). The CSES was originally developed to assess engagement in high school computer science courses. For this study, the survey was adapted to be appropriate for middle school students by simplifying the language and modifying some items to refer to the specific coding activities used in the curriculum.

The adapted CSES included items assessing behavioral, cognitive, and emotional engagement in computer science. Examples of items include "I pay attention in computer science class" (behavioral engagement), "I try to connect what I learn in computer science to my own experiences" (cognitive engagement), and "I enjoy the challenges of computer science assignments" (emotional engagement). Students responded to each item on a 5-point Likert scale ranging from "Strongly Disagree" to "Strongly Agree."

To ensure the validity of the adapted CSES, a panel of computer science education experts reviewed the modified items and provided feedback. The survey was also pilot tested with a small group of middle school students to check for comprehension and clarity. However, as this was the first use of the adapted CSES with middle school students, further validation is needed to establish its psychometric properties with this population.

Data Collection and Analysis

Quantitative Data

Pre- and post-intervention CTT and CSES data were collected from all participating students. Independent samples t-tests were used to compare the post-intervention scores between the pair programming and individual practice groups, controlling for pre-intervention scores.

Qualitative Data

A subsample of 24 students was purposely selected to participate in focus groups and interviews. These students were selected to represent a range of CT proficiency levels and demographic backgrounds. Semi-structured focus groups and interview protocols were used to explore students' experiences with pair programming or individual practice, their perceptions of the benefits and challenges, and how these activities influenced their CT skills and engagement.

Qualitative data were analyzed using thematic analysis (Braun & Clarke, 2006). The analysis process involved the following steps:

1. Familiarization with the data: Transcribing the focus groups and interviews, reading through the transcripts, and noting initial ideas.
2. Generating initial codes: Systematically coding interesting features of the data across the entire dataset.

3. Searching for themes: Collating codes into potential themes and gathering all relevant data for each theme.
4. Reviewing themes: Checking if the themes work in relation to the coded extracts and the entire dataset, and creating a thematic map.
5. Defining and naming themes: Refining the specifics of each theme and generating clear names and definitions.
6. Producing the report: Selecting vivid examples, relating the analysis back to the research questions and literature, and producing a scholarly report.

To enhance the trustworthiness of the qualitative findings, two researchers independently coded a subset of the data and then compared and reconciled their coding. Disagreements were resolved through discussion until consensus was reached.

Results

Quantitative Results

Independent samples t-tests were conducted to compare post-intervention CT skills and engagement between the pair programming and individual practice groups, controlling for pre-intervention scores. The results revealed significant differences in both outcome measures.

For CT skills, students in the pair programming group ($M = 85.6$, $SD = 10.2$) demonstrated significantly higher post-intervention CTT scores compared to those in the individual practice group ($M = 79.4$, $SD = 11.8$), $t(100) = 2.98$, $p = .004$, $d = 0.59$. This finding suggests that pair programming had a moderate positive effect on students' CT skills, as indicated by the effect size (Cohen's d).

Similarly, students in the pair programming group ($M = 4.2$, $SD = 0.6$) reported significantly higher levels of engagement compared to those in the individual practice group (M

= 3.8, SD = 0.7), t(100) = 3.31, p = .001, d = 0.66. This result indicates that pair programming had a moderate to large positive effect on student engagement, as evidenced by the effect size. Table 1 presents the means, standard deviations, t-test results, and effect sizes for both outcome measures.

Table 1

Means, Standard Deviations, and t-test Results for Post-Intervention CT Skills and Engagement

Variable	Pair Programming		Individual Practice		t	p	Cohen's d
	M	SD	M	SD			
CT Skills (CTT)	85.6	10.2	79.4	11.8	2.98	0.004	0.59
Engagement (CSES)	4.2	0.6	3.8	0.7	3.31	0.001	0.66

CT Skills measured by Computational Thinking Test (CTT). Engagement measured by adapted Computer Science Engagement Survey (CSES).

These results suggest that pair programming had a significant positive effect on both CT skills and engagement in introductory computer science classes.

Qualitative Results

Thematic analysis of focus group and interview data revealed several key themes related to students' experiences with pair programming and its perceived benefits and challenges. These themes are presented below, with illustrative quotes from participants.

Enhanced Collaboration and Peer Learning

Students in the pair programming group consistently reported that working with a partner facilitated collaboration and peer learning. They described how the "driver" and "navigator" roles allowed them to actively contribute to the problem-solving process and learn from each other's perspectives. For example, one student stated:

> "I liked working with a partner because we could bounce ideas off each other. When I was the driver, my partner would catch mistakes or suggest different ways to approach

the problem. When I was the navigator, I could learn from how my partner coded and ask questions if I didn't understand something" (Participant 12).

Increased Engagement and Motivation

Many students in the pair programming group expressed that collaborating with a partner made coding tasks more engaging and enjoyable compared to working independently. They reported feeling more motivated to persevere through challenges and complete projects when they had a partner to support and encourage them. As one student shared:

"Coding with a partner was way more fun than doing it alone. We celebrated together when we figured something out, and we kept each other going when things got tough. It was nice to have someone there to keep me motivated" (Participant 7).

Development of Problem-Solving Skills

Students in the pair programming group frequently described how collaborating with a partner helped them develop better problem-solving strategies. They reported that discussing problems, brainstorming solutions, and explaining their thought processes to their partner enhanced their ability to break down complex tasks, identify patterns, and design algorithms. One student explained:

"Working with a partner really helped me improve my problem-solving skills. We would talk through the problem together, come up with different ideas, and then test them out. I learned how to break things down into smaller steps and think more logically" (Participant 19).

Challenges with communication and coordination

While most students in the pair programming group reported positive experiences, some described initial challenges with effectively communicating and coordinating with their partner.

These challenges were often related to differences in working styles, skill levels, or personalities. However, many students noted that these difficulties decreased over time as they developed better communication strategies and learned to work together more smoothly. For instance, one student reflected:

> "At first, it was hard to get on the same page with my partner. We had different ideas about how to approach the problems, and sometimes we would get frustrated with each other. But as we kept working together, we got better at listening to each other's ideas and finding ways to compromise. By the end, we were a really good team" (Participant 22).

Overall, the qualitative findings suggest that pair programming provided valuable opportunities for collaboration, peer learning, and the development of problem-solving skills, while also increasing students' engagement and motivation in computer science learning. Although some students experienced initial challenges with communication and coordination, most reported that these difficulties diminished over time as they developed effective strategies for working with their partners.

Discussion & Implications

The quantitative results of this study provide evidence for the effectiveness of pair programming in improving CT skills and engagement among middle school students in introductory computer science classes. The significantly higher post-intervention CTT scores and engagement levels in the pair programming group compared to the individual practice group suggest that this collaborative approach can help reduce demographic gaps in these key outcomes. These findings align with prior research highlighting the potential benefits of pair programming for enhancing CT practices and engagement in computer science education (Denner et al., 2021; Werner et al., 2013).

The qualitative findings offer insights into the mechanisms through which pair programming may support CT skill development and engagement. Students' experiences suggest that collaboration, peer learning, and shared problem-solving are key factors that contribute to the effectiveness of pair programming. By working together, students can learn from each other, share knowledge, and collaboratively develop CT skills. Additionally, pair programming appears to increase engagement and motivation by making coding tasks more enjoyable and providing peer support to persist through challenges. These qualitative themes are consistent with previous studies highlighting the social and motivational benefits of pair programming (Williams & Kessler, 2003; McDowell et al., 2006).

The findings of this study have several implications for computer science education practice and research. First, the results support the use of pair programming as a culturally responsive pedagogical approach to promote inclusive and equitable computer science education in middle schools. By implementing pair programming activities, educators can help engage diverse learners, reduce demographic gaps in CT skills and engagement, and foster a more collaborative and supportive learning environment.

Second, the study highlights the importance of providing students with opportunities to develop communication and collaboration skills alongside technical coding abilities. While some students initially faced challenges with effectively communicating and coordinating with their partners, most reported that these difficulties diminished over time as they developed better strategies. Explicitly teaching and reinforcing communication and collaboration skills may help students navigate these challenges and maximize the benefits of pair programming.

Third, the findings suggest that pair programming can be effectively integrated into existing computer science curricula and class structures. The study demonstrates that significant

improvements in CT skills and engagement can be achieved by incorporating weekly pair programming activities while still covering the same content as individual practice. This suggests that pair programming can be a feasible and valuable addition to middle school computer science courses.

However, it is important to acknowledge the limitations of this study and the need for further research. The quasi-experimental design, while practical, may not fully control for potential confounding variables. Additionally, the study was conducted in a single school with a specific demographic composition, which may limit the generalizability of the findings to other contexts. Future research should replicate this study in diverse educational settings and employ more rigorous experimental designs to confirm the causal effects of pair programming on CT skills and engagement.

Furthermore, longitudinal studies are needed to investigate the long-term impacts of pair programming on students' academic and career trajectories in computer science. It would be valuable to examine whether the benefits of pair programming in middle school translate to continued engagement and success in higher levels of computer science education and beyond.

Conclusion

In conclusion, this mixed methods study provides evidence for the effectiveness of pair programming in improving computational thinking skills and engagement among diverse middle school students in introductory computer science classes. The quantitative results demonstrate that students who participated in weekly pair programming activities had significantly higher post-intervention CT skills and engagement levels compared to those who practiced individually. The qualitative findings suggest that collaboration, peer learning, and shared problem-solving are

key mechanisms through which pair programming supports CT skill development and engagement.

These findings contribute to the growing body of research on culturally responsive pedagogies in computer science education and offer practical implications for educators seeking to promote inclusive and equitable learning experiences. By incorporating pair programming activities into middle school computer science curricula, teachers can help reduce demographic gaps in CT skills and engagement, foster a more collaborative and supportive learning environment, and prepare diverse students for success in future computer science pursuits.

However, further research is needed to replicate these findings in other educational contexts, investigate the long-term impacts of pair programming, and explore additional strategies for supporting communication and collaboration skills in computer science classrooms. As we work towards a more inclusive and equitable future in computer science education and careers, identifying and implementing effective pedagogical approaches like pair programming will be essential for engaging and empowering all students to develop the computational thinking skills needed for success in our technology-driven world.

References

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology, 3*(2), 77-101.

Creswell, J. W., & Creswell, J. D. (2022). *Research Design* (6th ed.). SAGE Publications, Inc. (US).

Denner, J., Green, E., & Campe, S. (2021). Learning to program in middle school: How pair programming helps and hinders intrepid exploration. *Journal of the Learning Sciences*.

Google Inc. & Gallup Inc. (2016). *Diversity gaps in computer science: Exploring the underrepresentation of girls, Blacks and Hispanics*.

Jain, G., Martin, F., Feliciano, B., Hsu, H. Y., Fauvel-Campbell, B., Bausch, G., ... & Thomas-Cappello, E. (2022, October). CS Pathways: A Culturally Responsive Computer Science Curriculum for Middle School. In *2022 IEEE Frontiers in Education Conference (FIE)* (pp. 1-9). IEEE.

Madkins, T. C., Howard, N. R., & Freed, N. (2020). Engaging equity pedagogies in computer science learning environments. *Journal of Computer Science Integration, 3*(2), 1-27.

McDowell, C., Werner, L., Bullock, H. E., & Fernald, J. (2006). Pair programming improves student retention, confidence, and program quality. *Communications of the ACM, 49*(8), 90-95.

Ni, L., Bausch, G., Thomas-Cappello, E., Martin, F., & Feliciano, B. (2023, March). Creating Apps for Community and Social Good: Learning Outcomes of a Culturally Responsive Middle School Computer Science Curriculum. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (pp. 73-79).

Román-González, M., Pérez-González, J. C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior, 72*, 678-691.

Scott, A., Martin, A., McAlear, F., & Madkins, T. C. (2015). Broadening participation in computer science: Existing out-of-school initiatives and a case study. *ACM Inroads, 7*(4), 84-90.

Wang, M. T., Fredricks, J. A., Ye, F., Hofkens, T. L., & Linn, J. S. (2016). The Math and Science Engagement Scales: Scale development, validation, and psychometric properties. *Learning and Instruction, 43*, 16-26.

Werner, L., Denner, J., & Campe, S. (2013). Pair programming: Under what conditions is it advantageous for middle school students?. *Journal of Research on Technology in Education, 46*(3), 277-291.

Williams, L., & Kessler, R. (2003). *Pair programming illuminated*. Addison-Wesley.