PearProgram: Towards Fruitful Collaboration in Computing Education Research and the Learning Sciences

Jorge E. Garcia, Maxwell Bigman, Ethan Roy, Miroslav Suzara, Roy Pea
jorgeedu@stanford.edu, mbigman@stanford.edu, ethanroy@stanford.edu, msuzara@stanford.edu, roypea@stanford.edu
Stanford Graduate School of Education

Abstract. Computer Education Research has not made widespread use of relevant research in the Learning Sciences. Our research seeks to understand pair programming learning dynamics using established Learning Sciences frames to improve pedagogical methods and practices in Computer Science learning. We have developed *PearProgram* to work as both a learning tool and a data-gathering tool. Students who used the tool engaged more actively in the courses and attained higher pass rates. Additionally, analysis of qualitative data points to the importance of *Dual-Space Model* and *Positioning* frameworks in understanding the dynamics of a generative pair programming session. These findings suggest careful attention by the facilitator to offer structure and explicit decision making about roles can shape pair success in assignments and augment implicit learning.

Introduction

The emergence of the *Computer Science for All* movement brought research questions from Computing Education Research (CER)—the study of how people learn and teach computing—into adjacent education fields, including Learning Sciences (LS). Despite the emergence of best practices in computer science (CS) education addressing key research questions of how people can learn computing more effectively and equitably, the CER field may benefit from Learning Sciences research. Pair programming, a practice where two programmers work together on a single program, is an example of a CS education pedagogy with distinct roots in the collaborative learning literature, yet has not previously drawn from LS research, nor from computer supported collaborative learning (CSCL). Pair programming is considered a best practice in K-12 and university CS education, yet we have not found evidence of widespread utilization of relevant insights from the learning sciences used to inform how pair programming might leverage collaborative learning research.

In this paper, we present *PearProgram*, an online collaborative learning and research tool to help introductory CS students learn how to pair-program based on theory (for more, see www.pearprogram.com/#/about). We describe the theoretical foundation of the tool, promising results from early usage, and future research directions. We specifically highlight the potential importance of considering Positioning Theory when students work in (CS) collaborative learning settings. We aim to address the field between CER and LS research so that both may benefit from key findings on these discipline-specific collaborative learning practices that might advance new understandings for LS and CSCL researchers.

Pathways to pair programming insights

Pair programming is a K-12 and university best practice for CS education developed by professional software engineers in the '90s (Beck, 2000); its documented benefits for programmers of all levels has a long history (Williams et al., 2000). Twenty years of CS education research, including two rigorous meta-analyses, has demonstrated that "if implemented properly" pair programming has positive effects on students' programming assignment grades, exam scores, and persistence, especially in introductory CS classes (de Lima Salge & Berente, 2016; Umapathy & Ritzhaupt, 2017). Other benefits of pair programming include greater confidence (Hanks et al., 2004), deeper conceptual understanding (McDowell et al., 2006), and continued interest in the topic (Littleton & Howe, 2010; Werner et al., 2004). However, studies have documented that the benefits of pair programming are not always realized, suggesting unexplored phenomena may play a role.

Grounded by numerous studies on academic achievement showing students tend to learn better collaboratively than alone (Abrami & Chambers, 1996; O'Donnell & Dansereau, 1992), we nevertheless concur with Dillenbourg et al. (1996), who aver that: "collaboration is in itself neither efficient nor inefficient. Collaboration works under some conditions, and it is the aim of research to determine the conditions under which collaborative learning is efficient". Our aim in developing the *PearProgram* tool is to better understand the conditions that lead to pairs engaging in "efficient" (or generative) collaborative learning when pair programming, and how we might create the most fertile learning environments for all students learning CS.

Sparked by the global pandemic, learning experiences increasingly rely on computing tools and open up possibilities for understanding distributed learning. Our research, and the associated tool, *PearProgram*, is beginning to yield insights into how learning can unfold between pairs of distanced student learners. We were surprised to learn that our tool supported student engagement despite the isolation wrought by the pandemic. Our three aims in developing the *PearProgram* tool are: 1) To help introductory CS students learn how to pair program effectively; 2) To better understand how students pair program by capturing quantitative and qualitative data; 3) To research what factors lead to generative learning outcomes in different contexts.

PearProgram: A theory driven learning tool

Our understanding of pair programming is rooted in socio-constructivist theories of learning (Piaget 1926; Vygotsky 1978), underscoring the importance of social aspects of learning, with collaboration a primary means for thinking and learning. Our design choices stem from Barron's (2003) illumination of the complexities of collaborative work, theorized in terms of a "dual-space model" required for participants' collaborative problem solving, in which students need "to clarify how the content of the problem and the relational context are interdependent aspects of the collaborative situation." Prior CER work studied pair programming via elements of problem-solving processes (Williams et al., 2008), but has yet to focus on the corresponding and interdependent relational context. Our tool uses Barron's dual-space model to establish the importance of having students generate, confirm, document and reflect on one another's proposals so that pairs may more effectively collaborate. Research using our tool helps us understand how to support students navigating these complex spaces. To our knowledge, learning tools have not previously been designed to help students navigate these dual spaces in CS education. Since pair programming is a clear example of collaborative learning, our research aims to contribute to understanding what roles tools, behavioral nudges, and LS research can contribute to the complex task of such dual problem space problem solving.

Drawing on these theoretical foundations, the *PearProgram* tool allows us to establish turn-taking norms by designating specific roles for students (as pilot and co-pilot) with embedded role-changing prompts within a collaborative text editor. The tool is designed to be a student's first exposure to pair programming, inculcating collaborative practices that help students navigate the dual problem space of learning how to program collaboratively. Students are given equitable access to problem-solving processes by explicitly stating in the tool that "mutual understanding is *the* learning goal." While the implicit learning goal for students when pair programming typically is that they complete the programming task, individually learning the key concepts related to the task at hand, in *PearProgram*, we layer a new learning goal: that students help one another understand every line of code. Moreover, we draw on research suggesting that effective collaborations require that students view their success as mutually dependent; this leads to dialog that fosters new, co-created ideas and works to establish a shared social reality (Rogoff, 1990; Roschelle & Teasley, 1995). Accordingly, the tool provides embedded tips which emphasize that success *and* learning of the pair is a mutually dependent process.

Additionally, our tool emphasizes questions and annotations of the code as key mechanisms for driving learning. *PearProgram* has buttons for questions and comments to encourage students to ask questions and prompts students to establish joint attention (Tomasello, 1995). Building on Schneider and Pea's (2013) understanding of joint attention as both physically and verbally constructed, the *PearProgram* tool makes use of the programming environment to visually highlight lines of code when students ask their partner a question. Utilizing visual synchronization in the tool, we theorize that there will be better coordinated joint attention, which will increase the quality of collaboration and learning. Moreover, by actively engaging in learning-by-teaching (Goodland & Hirst, 1989), we anticipate mutual benefit for students using the tool.

Our design similarly draws on the "dual space model" as a mechanism for research analysis as we capture data that illuminates joint meaning-making activity during pair programming learning sessions. In particular, we draw on the dimensions of Meier, Spada and Rummel's (2007) framework for assessing the quality of computer-supported collaboration processes as a way to analyze our pair programmers; these include "sustaining mutual understanding, dialogue management, information pooling, reaching consensus, task division, time management, technical coordination, reciprocal interaction, and individual task orientation" (p. 63). As we analyze our data, these dimensions offer key metrics for understanding what activities effective pairs engage in that less effective dyads do not. Based on our preliminary analysis, these nine important dimensions nonetheless elide an important aspect of collaborative interaction: *Positioning*.

Positioning in pair programming

Davies and Harré's (1990) Positioning Theory offers an analytic lens and an explanatory theory for how discourse interacts with learning and identity development: "as an explanatory theory, Positioning Theory serves as a set of guiding principles for investigating the consequences of the discourse and the interactions of, and with, particular students and groups of students as they assume or reject particular positions or acts of positioning" (Green et. al , 2020). As a lens for understanding pair programming interactions, Positioning Theory offers a conceptually-driven approach to understand positions, acts, and storylines (Davies & Harré, 1990; Harré & van Langenhove, 1999). Incorporating Positioning Theory as an analytic tool in the "dual space model" of collaborative problem solving may reveal new insights into how students navigate the relational elements of collaborative work. Positioning Theory is useful as we center questions of power and privilege in the Learning Sciences and in CS (Annamma & Booker, 2020; Vakil, 2018). It is essential to move toward justice-centered approaches to understanding equity in CS and collaborative learning.

Preliminary findings

PearProgram for in-class activities in three remote introductory CS classes has led to results suggesting the tool has purchase for advancing student learning and research insights. Our post-use survey for instructors indicated that all instructors felt the tool positively contributed to the course; likewise students' post-use survey indicated they had improved as coders and wanted to engage further with *PearProgram*. Further, students who used the tool engaged more actively in the online class discussion forum and attained higher pass rates. While these findings are correlational and cannot ground causal inference, they suggest merits for further investigating the conditions contributing to efficacious uses of *PearProgram*. Moreover, our initial qualitative data illuminates a fertile research topic: positioning in pair programming and collaborative learning. Video analyses of pair programming interactions and student interviews surface findings that contribute to both CER and CSCL fields.

Beyond the quantitative data described, we video-recorded pair work involving lists in Python and conducted some post-use interviews. We focused on one pair who finished their task in ⅓ the time of the others and engaged in unique behaviours compared to other groups. This pair centered the conversation of roles (pilot/co-pilot) in their first ½ minute of interaction. In this interaction, we conjecture that relational work was being accomplished in positioning one another as peer learners, and accruing early practices in joint decision making revealed that after this initial decision, talk time was evenly distributed between the learners throughout the rest of the session compared to other pairs, suggesting the value of this interaction:

Student 1:	So do you want to be pilot or co-pilot? I'm cool with whatever.
Student 2:	Um, right now it says I'm co-pilot. Does it say you're pilot?
Student 1:	Yeah, but if you wanna switch that's like cool with me too. It's up to you.
Student 2:	Um, maybe we can switch like halfway or something? I don't know.
Student 1:	Ok. Yeah that works. Ok. um.

Future study designs

As we design future studies, we use the insights of Positioning Theory to investigate unexamined dimensions of collaborative learning that may prove crucial for understanding the nature of computer-supported collaborative learning environments for students, particularly in the CS domain, fraught with positioning challenges (e.g., stereotype threat). As we progress with the *PearProgram* project, we are conducting contrasting case analyses, comparing successes and difficulties with in-person, remote and hybrid pair programming, as we look for 'positive deviance'—"intentional behaviors that depart from the norms of a referent group in honorable ways" (Spreitzer & Sonenshein, 2004). As learning environments are increasingly technology-mediated and span remote settings in the Covid-19 era, careful attention to the learning tools, the metrics of success, the complexities of navigating collaborative learning environments, including relational spaces, takes on added significance. For educational tools, we adopt the approach that technology should create experiences that go "beyond being there" (Hollan & Stornetta, 1992)—establishing learning opportunities uniquely possible in online settings. To that end, we believe we are also creating research opportunities that can go "beyond being there".

References

Abrami, P. C., & Chambers, B. (1996). Research on cooperative learning and achievement: Comments on Slavin. *Contemporary Educational Psychology*, 21(1), 70-79.

Annamma, S.A. & Booker, A. (2020). Integrating intersectionality into the study of learning. In N. Nasir, C. Lee, R. Pea, & M.M. De Royston. (Eds.). *Handbook of the cultural foundations of learning* (pp. 297-313). New York: Routledge.

Barron, B. (2003). When smart groups fail. *The Journal of the Learning Sciences*, 12(3), 307–359.

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

Bigman, M., Roy, E., Garcia, J. E., Suzara, M., Wang, K., Piech, C. (2021, in press). PearProgram: A more fruitful approach to pair programming. In Proceedings of the *52nd ACM Technical Symposium on Computer Science Education* (SIGCSE '21).

Davies, B., & Harré, R. (1990). Positioning: The discursive production of selves. *Journal for the Theory of Social Behaviour*, 20(1), 43–63.

de Lima Salge, C. A., & Berente, N. (2016). Pair programming vs. solo programming: what do we know after 15 years of research? In 2016 *49th Hawaii International Conference on System Sciences* (HICSS) (pp. 5398-5406). Piscataway, NJ: IEEE.

Dillenbourg, P., Baker, M., Blaye, A., & O'Malley, C. (1996). The evolution of research on collaborative learning. In P. Reimann & H. Spada (Eds.), *Learning in humans and machine: Towards an interdisciplinary learning science* (pp. 189–211). Oxford: Elsevier.

Green, J.L., Brock, C., Baker, W.D., & Harris, P. (2020). Positioning theory and discourse analysis: an explanatory theory and analytic lens. In N. Nasir, C. Lee, R. Pea, & M.M. De Royston. (Eds.). *Handbook of the cultural foundations of learning* (pp. 119-140). New York: Routledge.

Goodlad, S., & Hirst, B. (1989). *Peer tutoring. A guide to learning by teaching*. New York: Nichols Publishing.

Hanks, B., McDowell, C., Draper, D., & Krnjajic, M. (2004). Program quality with pair programming in CS1. In Proceedings of the *9th annual SIGCSE conference on innovation and technology in computer science education* (pp. 176-180).

Harré, R., & van Langenhove, L. (1999). *Positioning theory*. Oxford: Blackwell.

Hollan, J., & Stornetta, S. (1992). Beyond being there. *SIGCHI Proceedings* (pp. 119-125).

Littleton, K., & Howe, C. (Eds.). (2010). *Educational dialogues: Understanding and promoting productive interaction*. New York: Routledge.

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.

Meier, A., Spada, H., & Rummel, N. (2007). A rating scheme for assessing the quality of computer-supported collaboration processes. *International Journal of Computer-Supported Collaborative Learning*, 2(1), 63–86.

O'Donnell, A. M., & Dansereau, D. F. (1992). Scripted cooperation in student dyads: A method for analyzing and enhancing academic learning and performance. In N. Miller & R. Hertz-Lazarowitz (Eds.), *Interaction in cooperative groups: The theoretical anatomy of group learning* (pp. 120-141). New York: Cambridge University Press.

Piaget, J. (1926). *The language and thought of the child*. London: Routledge & Kagan Paul.

Rogoff, B. (1990). *Apprenticeship in thinking: Cognitive development in social context*. Oxford: Oxford University Press.

Roschelle, J, & Teasley, S.D. (1995). The construction of shared knowledge in collaborative problem solving. In C. O'Malley (Ed.), NATO ASI Series,Vol. 128: *Computer supported collaborative learning*. Berlin, Heidelberg: Springer.

Schneider, B., & Pea, R. (2013). Real-time mutual gaze perception enhances collaborative learning and collaboration quality. *International Journal of Computer-supported collaborative learning*, 8(4), 375-397.

Spreitzer, G. M., & Sonenshein, S. (2004). Toward the construct definition of positive deviance. *American Behavioral Scientist*, 47(6), 828-847.

Tomasello, M. (1995). Joint attention as social cognition. In C. Moore & P. J. Dunham (Eds.), *Joint attention: Its origins and role in development* (pp. 103–130). Hillsdale, NJ: Erlbaum Associates.

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* (TOCE), 17(4), 1-13.

Vakil, S. (2018). Ethics, identity, and political vision: Toward a justice-centered approach to equity in computer science education. *Harvard Educational Review*, 88(1), 26–52.

Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes*. Cambridge: Harvard University Press.

Werner, L. L., Hanks, B., & McDowell, C. (2004). Pair-programming helps female computer science students. *Journal on Educational Resources in Computing* (JERIC), 4(1), 4-es.

Williams, L., Kessler, R. R., Cunningham, W., & Jeffries, R. (2000). Strengthening the case for pair programming. *IEEE software*, 17(4), 19-25.

Williams, L., McCrickard, D. S., Layman, L., & Hussein, K. (2008). Eleven guidelines for implementing pair programming in the classroom. In *Agile 2008 Conference* (pp. 445-452). IEEE.