Evaluating the Efficacy and Impacts of Remote Pair Programming for Introductory Computer Science Students

Mustafa Hafeez, Anand Karki, Yara Radwan, Anis Saha,
Angela Zavaleta Bernuy, Brian Harrington
Dept of Computer and Mathematical Sciences
University of Toronto Scarborough

Introduction

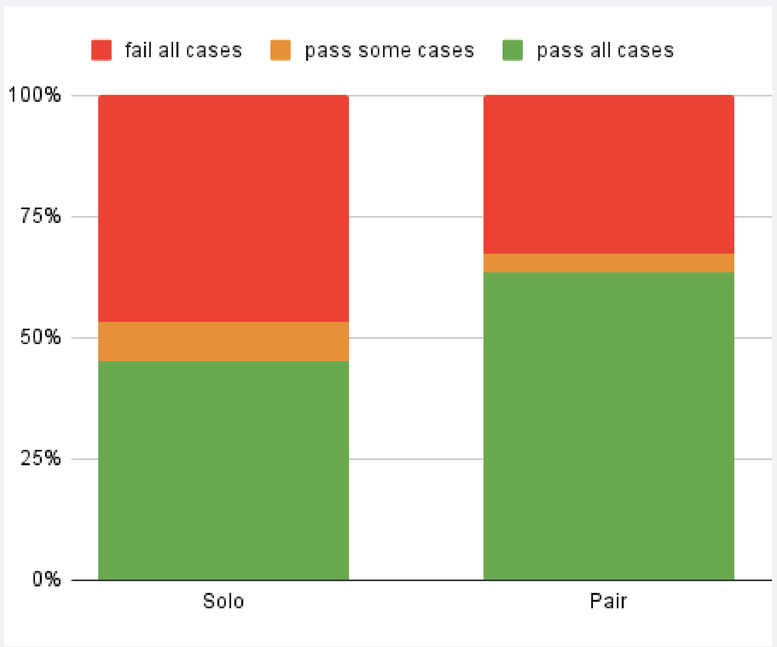
Prior research has shown that integrating pair programming in introductory computing courses encourages student interaction, improves communication skills, and enhances comprehension of technical material, problem-solving, and software proficiency. This paper replicates existing studies in pair programming in an online context.

This study replicates existing work on the impact of pair programming on program correctness and complexity, and time taken to write code [1], [2]; students' perception of code including difficulty [3], speed[4], and confidence[2], [4], [5]; and students perceptions of pair programming as a methodology [1], [3], [4]

Methodology

- Fall 2021 - CS0 course for non-majors - CSCA20: Introduction to Programming
- 116 students randomly assigned to either Solo or Pair treatment
- 4 weeks, 30min sessions, complete programming question + survey, conducted over Zoom
- Students in pair condition: following 'driver navigator' methodology [6]

Correctness



- Pair and solo treatments had no difference in time taken to complete task or number of attempts made
- No pattern found in code complexity as measured by logical lines of code (LLOC)
- Pair treatment groups were more likely to produce working solutions, with a lower probability of errors in final submissions
- Pair treatment groups had lower incidences of runtime errors, as predicted
- After Bonferroni correction for multiple tests, results could not be shown to be statistically significant

Perceptions

- Students in the pair programming condition reported significantly higher confidence in their solutions ($p < 0.0125$).
- Students in the pair programming condition were in strong agreement that pair programming helped them find and repair more errors ($p < 0.001$).
- Students in the pair programming condition felt that pair programming helped them finish their work more efficiently ($p \leq 0.01$).

Statistical Analysis

	Solo Mean	Pair Mean	Solo SD	Pair SD	p-value
Time Spent (Minutes)	4876.180	4088.346	7444.154	4759.596	0.407
Logical Lines of Code (LLOC)	17.592	18.307	4.628	4.422	0.339
Correctness	0.496	0.697	0.471	0.430	**0.02***
Runtime error rate	0.220	0.101	0.406	0.289	**0.0301***
Compile error rate	0.141	0.057	0.350	0.235	0.068
Num Attempts	7.708	6.670	7.961	5.512	0.328
(Survey) Difficulty	6.119	6.108	2.549	2.670	0.907
(Survey) Confidence in Solution	3.387	3.928	1.595	1.296	**0.0115****
(Survey) Speed Perception	0.645	0.788	0.48	0.41	**0.01****
(Survey) Error fix perception	2.924	4.034	1.123	0.969	**<0.0001****
(Survey) Solo/Pair Code Better	3.488	4.104	0.862	0.981	**<0.0001****

Comparison of outcomes of solo vs pair conditions, showing statistically significant results
(* indicates statistically significance at $p < 0.05$),
(** indicates statistically significance at the corrected threshold of $p < 0.0125$)

Conclusions

- We were able to replicate multiple pair programming studies that were conducted in-person, in a remote context to see which findings were replicated.
- Pair programming in an online setting appeared to have a positive—but not statistically significant—impact on code correctness, but no impact on speed or complexity.
- Online pair programming had similar positive impacts with respect to student perceptions, except in the case of perceived difficulty where no difference was found between conditions.
- Pair programming using a driver navigator model is possible online, but further research is needed to determine which benefits and drawbacks transfer to an online environment.

References

[1] A. Cockburn and L. Williams, "The costs and benefits of pair programming," in *Extreme programming examined*, 2001, pp. 223–243.

[2] C. McDowell, L. Werner, H. E. Bullock, and J. Fernald, "Pair programming improves student retention, confidence, and program quality," *Communications of the ACM*, vol. 49, no. 8, pp. 90–95, Aug. 2006.

[3] N. Nagappan, L. Williams, M. Ferzli, *et al.*, "Improving the cs1 experience with pair programming," *ACM Sigcse Bulletin*, vol. 35, no. 1, pp. 359–362, 2003.

[4] S. Faja, "Evaluating Effectiveness of Pair Programming as a Teaching Tool in Programming Courses," en, p. 10, 2014.

[5] B. Hanks, "Student performance in cs1 with distributed pair programming," *ACM SIGCSE Bulletin*, vol. 37, no. 3, pp. 316–320, 2005.

[6] L. A. Williams and R. R. Kessler, "All I really need to know about pair programming I learned in kindergarten," *Communications of the ACM*, vol. 43, no. 5, pp. 108–114, May 2000.