Toward Adaptive Collaborative Support for Elementary Students Learning Computer Science

Jennifer Tsan
North Carolina State University
jtsan@ncsu.edu
717-318-7836
Raleigh, NC

Advisors:
Collin F. Lynch, North Carolina State University
Kristy Elizabeth Boyer, University of Florida

Toward Adaptive Collaborative Support for Elementary Students Learning Computer Science

Jennifer Tsan, North Carolina State University, jtsan@ncsu.edu
Collin F. Lynch, North Carolina State University, cflynch@ncsu.edu
Kristy Elizabeth Boyer, University of Florida, keboyer@ufl.edu

Abstract: Collaboration is an important aspect of learning. For computer science, pair programming has been shown especially beneficial. I have begun to study pair programming dialogue with elementary school students doing block-based programming. My preliminary results show that students can struggle to engage in balanced collaborations. This problem is in part due to difficulty sharing the controls, a lack of understanding of pair programming roles, and the need to build good collaborative dialogue practices. For my dissertation, I propose to develop and iteratively refine a collaborative block-based programming environment to support real-time collaboration for young students.

Keywords: K-12, pair programming, real-time collaboration

Goals of Research

The goal of this research is to gain insight into how young children pair program and how we can better support them. Pair programming is a method in which two programmers work side by side. One programmer is the driver who has control of the keyboard and mouse and does the coding. The other programmer is the navigator, who is to actively watch the driver program, look for mistakes, think ahead to solve the problem, and ask questions. Programmers generally switch roles after an allotted amount of time or after a task is completed. Studies using pair programming indicate that it leads to increased confidence in students, a higher course completion rate for undergraduate students, and higher quality code (Braught, Wahls & Eby 2011; Cockburn & Williams, 2000; McDowell, Werner, Bullock & Fernald, 2002).

Research questions

My overarching research question is, *How can we improve current programming environments to adaptively support elementary students in pair programming?* To investigate that question I will also address the question, *How can we build adaptive programming environments to support good collaborative dialogue practices?*

Current Status and Preliminary Results

I collected data from a computer science elective in an elementary school, including videos of students pair programming. Research questions I have started to investigate are: *How do young coders balance their dialogue, turn-taking and control during collaborative computer science learning?*; and *How do young learners coordinate their dialogue during collaboration for computer science?* I found that elementary students are often unbalanced in terms of how much they speak, drive, and contribute ideas to the project. The imbalance in the pair programming relationships may be due to students having difficulty understanding their roles and not knowing good dialogue practices for collaboration. Therefore, to support the students, we should teach them good pair programming and collaborative dialogue skills. These skills include: staying active in either role, with both students contributing suggestions throughout the process and the navigator asking more questions; sharing, not only the keyboard and mouse, but also the responsibilities of their roles; and building upon each other's ideas.

Plan

After identifying problems and potential support points for young students pair programming, I would like to modify existing programming environments such as Scratch to better support their pair programming process. I specifically plan to help students to share and stay active in their roles. When pair programming on one computer, it may be difficult for students to know how to share physical controls, even when they are assigned roles. Therefore, it may be helpful to modify the environment to allow multiple students to login using different computers to view and modify the same project. Such collaborative tools are available for textual programming (Tran, et al., 2013; Goldman, Little & Miller, 2011), however, few are available for young children (Al-Jarrah & Pontelli, 2014).

The modified programming environment will allow the students to work on the same project on separate computers. We will consider a number of design decisions based on user studies and iterative refinement, as well as a literature review of existing collaborative tools for young children and broader audiences. These design considerations include:

- How should the synchronized collaborative support be designed?
    - Should both students be able to edit at the same time, similar to how collaborators can work in Google docs?
    - Should we limit it to only person editing at once? The work will synchronize and both partners will be able to see the changes and they can switch controls at any time, but the partner who isn't editing cannot make changes.
    - Should we limit it to only the driver editing? The work will synchronize and both partners can see the changes. After an allotted amount of time (i.e., 10 minutes) the software could have the students switch roles and the current driver will be the only one that can make edits.
- What types of messages should be delivered, and when, to support effective collaboration for computer science problem solving?

Issues and Problems for Further Discussion

At the doctoral consortium, I would like to receive feedback on my research plan. After a literature review and initial pilot, I plan to iteratively test and refine the software. For the studies, I will recruit elementary students who are participants of clubs or classes that involve programming, technology, or computer science. The students will be given 1 hour of instruction on the environment, take individual pre-tests, and then they will pair program to solve a problem using the modified version of the coding environment. Afterwards, they will take individual post-tests and surveys, then I will hold 30-minute focus groups to obtain feedback on how well the software ran and supported the pair programming process. Then studies will have two conditions: students pair programming on one computer using the original coding environment; and students pair programming on two computers using the modified coding environment. These conditions will help me determine the ways in which the features I add support student collaboration.

After each study, I will analyze the students' dialogue and actions to determine whether students in the experiment condition used better collaborative dialogue practices and fulfilled each role better. In addition, I will calculate the students' learning gains to determine whether students in a specific condition benefited more from their collaboration. Between each study, I will refine the software based on the findings from the data.

Expected contributions

By the end of my dissertation I would like to have contributed methods and a tool to further support young students collaboratively solving programming problems. I hope my work will enlighten the community on how we can better support collaboration between young students in the future.

References

Al-Jarrah, A., & Pontelli, E. (2014). " AliCe-ViLlagE" Alice as a Collaborative Virtual Learning Environment. In *Frontiers in Education Conference (FIE), 2014 IEEE* (pp. 1-9).

Braught, G., Wahls, T., & Eby, L. M. (2011). The case for pair programming in the computer science classroom. *ACM Transactions on Computing Education (TOCE)*, *11*(1), 2.

Cockburn, A., & Williams, L. (2000). The costs and benefits of pair programming. *Extreme programming examined*, 223-247.

Goldman, M., Little, G., & Miller, R. C. (2011). Collabode: collaborative coding in the browser. In *Proceedings of the 4th international workshop on Cooperative and human aspects of software engineering* (pp. 65-68).

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.

Tran, H. T., Dang, H. H., Do, K. N., Tran, T. D., & Nguyen, V. (2013). An interactive Web-based IDE towards teaching and learning in programming courses. In *Teaching, Assessment and Learning for Engineering (TALE), 2013 IEEE International Conference on* (pp. 439-444).

Acknowledgments

This work is supported in part by Google through a CS Capacity Research Award.