This is a preprint of the final chapter published in Hrastinski, S. (Ed.) (2021). Designing Courses with Digital Technologies: Evidence and Examples by and for Higher Education Instructors. Routledge.

Online Pair Programming

David Parsons, The Mind Lab

Darcy Vo, The Mind Lab

Karen Lambrechts, The Mind Lab

Abstract

This chapter describes a digital learning activity for developing coding skills that uses online pair programming, in which the participants perform coding tasks in pairs, thus collaboratively supporting each other's learning. The activity involves a series of coding sessions in which the participants rotate through different pairs, taking on the alternating roles of "driver" and "navigator", working on tasks that build upon each other. This pair rotation ensures that learning is distributed across the class. The online delivery is facilitated via an online meeting tool that enables breakout rooms and screen sharing. The programming language used in the example is Scratch, but other coding tools could also be used. Evaluation of the activity, using quantitative and qualitative survey data, indicates that the design of the online learning activity is effective in developing coding skills in an environment that promotes collaborative learning.

Background

The context of this digital learning activity is a postgraduate program in future-focused pedagogy for in-service teachers, delivered fully online. Part of this program focuses on digital technology areas of the curriculum, including computational thinking and the creation of code using the fundamental structures of sequence, selection and iteration. The digital learning activity described in this chapter is designed to meet these requirements though an online pair programming approach.

Pair programming is a technique from agile software development, where coding is done in pairs. One coder takes the role of driver (has control of the keyboard) while the other takes the role of navigator (observes and advises). These roles are regularly switched within the pair and the membership of the pairs is also changed regularly. In software development, pair programming can provide "improved team discipline, cohesion and morale, better code, a more resilient workflow and creation of better solutions." (Parsons, Ryu & Lal, 2008). Pair programming has been used extensively in education as a way of teaching programming skills to students. Research suggests that it can increase student satisfaction and improve student achievement as measured by grades (Salleh et al 2010). It has also been suggested that female students can gain particular benefits from pair programming (Hanks et al., 2011).

Pair programming has traditionally been a co-located activity, but with the increasing use of online learning, educators need to consider ways of gaining the benefits of pair programming in online contexts. Jun et al. (2007) suggested that online pair programming leads to more collaborative behavior than co-located pairing. However, some other studies of online pair programming have provided varying results, indicating a need for further research (Faja, 2011).

The learning activity described here takes account of this background in its design. It provides a template for running pair programming activities online by using a web-based meeting tool that supports breakout room and screen sharing functions. We used Zoom Meetings, but other tools (such as Google Meet) have similar features.

Digital learning activity

This digital learning activity is based on the idea that "knowledge is constructed by learners through and with others" (Salmon, n.d.). In this spirit, learners work together in rotating pairs in a synchronous online learning activity. By the end of the activity, the aim is that the learners will have increased their understanding of computational thinking concepts and see the value of using pair programming to learn about coding.

When developing this activity, we considered the constraints of the online learning environment in which the learners would not physically sit alongside each other. To address this, we utilized the breakout rooms feature of the online meeting tool to divide the learners into pairs in their own virtual rooms where they could use the screen sharing function to work together.

When we have run this activity, we have used the Scratch programming language, and the activities have related to graphics programming (drawing lines and shapes). However other programming languages and activities may be used as long as they follow the steps through the computational thinking skills introduced.

Task summary: Learners are put into breakout rooms in pairs in order to complete a series of coding exercises. The specifics of these exercises are outlined in the "activity structure" (below). One member of the pair will be the "driver" (creating the code while sharing their screen) and the other will be the "navigator" (reviewing and suggesting). The navigator also copies the code from the driver's screen so both driver and navigator have the same code when they change roles and partners. From time to time while the learners are in the breakout rooms, they will receive a message to change roles in the pair. The breakout rooms also need to be changed between activities, so the pairs are reformed with new partners.

Materials: The main tool required, apart from the online meeting tool, is a programming platform that is appropriate to the level pf experience of the learners. We used Scratch, which is appropriate to those who are just beginning to learn about coding. To frame the purpose of the activity, it is also helpful for learners to watch video or read an article that signifies the importance of learning to code, with an emphasis on creativity and digital fluency. The choice of material would depend on the age of the learners, for example Linda Liukas' book "Hello Ruby" for the youngest students, Mitch Resnick's Ted Talk "Let's Teach Kids to Code" for older students, or Jeanette Wing's article on "Computational Thinking" for more advanced students.

Activity structure: Before the learners are put in the breakout rooms for each coding session, the instructor demonstrates the next task, each of which is based on a different coding structure. When completing the pair programming activities, the learners can either be sharing their screen (driver) or can review and suggest (navigator) depending on their role. Once the learners are in their pairs in breakout rooms they co-create the code, find solutions and troubleshoot errors. Between coding sessions, they are invited to share what they have done with others in the main virtual room.

There are four pair programming stages in the activity, based on fundamental algorithmic constructs. These are:

1. Sequence: The first task is to write code that uses a sequence of steps to create graphics output on the screen (e.g. one or more straight lines).
2. Iteration: The second task is to write code to draw three different regular shapes using lines and loops (iteration). Suggested shapes are squares, equilateral triangles, and circles.
3. Selection: The third task is to create code to simulate the tossing of a coin (to show either "heads" or "tails") using a random number generator. A follow up extension activity is to code similar examples that have more than two options (e.g. "rock, "paper", "scissors" or the numbers on a die).
4. Nested Iteration: The fourth and final task is to create code that uses nested loops to create interesting patterns based on the shapes created in stage 2.

The role of the facilitator during the session is to provide live online support where requested, to indicate to learners when they should swap roles, and to facilitate the introductions (including demonstrations of key concepts) to each coding round.

Schedule and time: The activity takes about 2 hours overall. A suggested schedule is:

- Introduction to the session and the concepts of pair programming (15 minutes)
- First pair programming stage (sequence) – including introductory demo and feedback (20 minutes)
- Second pair programming stage (iteration) – including introductory demo and feedback (30 minutes)
- Third pair programming stage (selection) – including introductory demo and feedback (15 minutes)
- Fourth pair programming stage (nested iteration) – including introductory demo and feedback (30 minutes)
- Summary (10 minutes)

Note that pairs should be changed before each of the four stages. Roles should be changed about every 5 minutes during each stage.

Follow up activity: This activity covers the basics of sequence, iteration and selection. The next suggested activity would be a follow-up pair programming session that builds on these skills to integrate the iteration and selection aspects of coding. This would require introducing and explaining the use of variables.

Evaluation

When evaluating the digital learning activity, we focused on two questions:

How can pair programming be effectively delivered online in an educational context?

How do in-service teachers perceive the value of pair programming as a technique for teaching digital curricula?

The first question relates to the learning design, and the second to the student experience. Data were gathered through voluntary and anonymous questionnaires administered at the end of the activity, which included both quantitative and qualitative questions. We received 67 responses out of a total of 72 students enrolled on our postgraduate programme for in-service teachers.

To answer the first question, we asked participants how much the session had developed their skills, and whether they would use the same type of session with their students. By the end of the session, 39 out of 67 participants felt that they had significantly developed their coding skills. Only one person felt that they had not developed their skills at all, as they had already used Scratch extensively. One participant commented that "*having different partners also gave a diverse spread of knowledge*". In terms of making pair programming effective, participants suggested ideas relating to the structure both as a group, "*you would need to have a real shared understanding within the class of working together norms*", and structure within pairs such as allowing students to change roles when they were ready rather than on command.

We asked participants "How likely are you to use this activity with your own students?" 33 out of the 58 who responded to this question agreed with the statement that they would be very likely to do this. We gave participants the option to skip this question if they felt it was not applicable to their context. This reinforces that this is an effective way to deliver pair programming in an educational context. However, one comment from a participant was "*I would like to incorporate this into my teaching, however I will need to reimagine how I could do this?*" This raises questions about how the activity might translate into other contexts using other tools. We address this question in the conclusions section.

Participants overall felt that pair programming was effectively delivered online in an educational context. This was shown in the data by the fact that many felt they had significantly developed their coding skills during the session, and they would be likely to use this activity with their own students.

Regarding the second research question, the learners had a positive perception towards pair programming as a technique, with 37 out of 67 agreeing with the statement that they would prefer pair programming than coding on their own. Another 22 were neutral on this question. Most of the learners, 54 out of 67, considered it valuable as a technique for teaching digital curricula that may be different to what they were used to, for example "*It was nice to look at more simple ways to introduce it to the classroom".*

Specifically, the collaborative aspect of pair programming was seen to have provided a supportive learning experience for learners, since it allows the learners to learn from each other. This was particularly evident where there were differences in skills between members of the pair; "*I really like the way we were able to help each other. Some were knowledgeable which was helpful, some felt the same as me, less confident. That was great from both sides*". Collaborative learning showed value in skills development; "*Great experience to learn and debug together!*" Overall, the teachers' perceptions were that pair programming as a learning technique was enjoyable, effective, and promoted collaboration.

Conclusions and future development

Having delivered this workshop several times and gained feedback from the participants, we believe that it is possible to effectively deliver a pair programming activity online using a suitable video-based tool that enables individual breakout rooms to be set up. Importantly, we believe that the intended value of pair programming as a learning technique, which emphasizes collaboration, mutual support, and engagement, can be maintained in a fully online context,

Delivering online pair programming is not, however, without its challenges. There are technical issues that may make it hard for students to work effectively with online programming tools in the video meeting context, and student feedback has also raised problems with skill alignment and the tendency of some students to opt out by leaving the virtual classroom.

For future development, we note that programming tools vary, which means the learning activity would have to be redesigned in various ways to cater for different programming tools. Rotating pairs is the greatest challenge. Although there are many collaborative programming tools available, these are designed so that multiple people can work on the same source code. It is harder to arrange for students to switch virtual pairs and carry their previous work with them. However, there may be other approaches that we can take to explore alternative online pair programming strategies.

In our workshops we have been using Scratch. One way of transferring code from one pair to another would be to share the code from each pairing session as a public project that could be copied and "remixed" by the next pair. We might also reconsider the way in which we carry code across from one coding session to the next. When software developers pair, they are working on a common code base. In these sessions, that is not the case. There are many separate code bases doing much the same thing. When students switch pairs, they often have two very different versions of the ongoing solution from which they must decide the elements they will use going forward to the next step of the activity. Since the idea of pair programming in this context is to learn from others, it is not necessary that either member of the pair carries their own code with them to the next coding stage. Instead, a series of collaborative coding accounts could be set up that students log in and out of between each pairing session.

In summary, we believe that our experience with designing and delivering an online pair programming activity demonstrates its value for learning to code in a collaborative digital environment. Future development will address follow up activities and alternative digital platforms for coding and collaborative online learning.

References

Faja, S. (2011). Pair programming as a team based learning activity: a review of research. *Issues in Information Systems, 12*(2), 207-216.

Jun, S., Kim, S., & Lee, W. (2007). Online pair-programming for learning programming of novices. *WSEAS Transactions on Advances in Engineering Education, 4*(9), 187-192.

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.

Parsons, D., Ryu, H. & Lal R. (2008, December). Better, Not More Expensive, Faster? The Perceived Effects of Pair Programming in Survey Data. In S. Huff & A. Mills (Eds.), *Creating the Future: Transforming Research into Practice – Proceedings of 19th Australasian Conference on Information Systems (ACIS 2008)* (pp. 710-719). AIS.

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.

Salmon, G. (n.d.) *E-tivities-Introduction*. https://www.gillysalmon.com/e-tivities.html