Remote pair programming

Hughes, Janet; Walshe, Ann; Law, Bobby; Murphy, Brendan

Published in:
Proceedings of the 12th International Conference on Computer Supported Education (Volume 2)

DOI:
10.5220/0009582904760483

Publication date:
2020

Document Version
Peer reviewed version

Remote Pair Programming

Janet Hughes[a], Ann Walshe, Bobby Law[b] and Brendan Murphy
School of Computing & Communications, The Open University, Walton Hall, Milton Keynes, U.K.
{*Janet.Hughes, Ann.Walshe, R.Law, B.Murphy*}*@open.ac.uk*

Keywords: Pair Programming, Employability, Distance Learning.

Abstract: Computing students often learn to program individually or in variously-sized groups whilst studying in computing laboratories and face-to-face classes. Previous research indicates that learning via pair programming can lead to students improving the quality of their programming, enhancing their programming skills and increasing their self-confidence when programming. Pair programming also is well established as a mechanism that supports peer learning and self-assessment for novice and more experienced students of programming. Observed benefits include increased self-efficacy, sharing of expertise, improved communication and teamworking – all enhancing employability. A considerable amount of existing work has examined pair programming benefits as they relate to campus-based students pairing face-to-face in a laboratory class – but how can distance learning students experience such benefits? This paper describes the preliminary results from a pilot study to investigate the benefits to distance learning students of engaging in Remote Pair Programming in their learning. Our investigation goes beyond academic learning to explore community, social and employability benefits, all of which are relevant to national measures of student satisfaction.

1 INTRODUCTION

> "With almost 200,000 registered students, the OU is the largest academic institution in the UK and, apart from a small number of postgraduate research students, most OU students study off-campus. It has a network of more than 5,000 tutors and uses communications technology to deliver teaching and assessment." (THE, 2020)

In October 2019, students registered to study approximately 19,000 modules in the School of Computing & Communications at The Open University (OU) in the UK. We are aware that traditional campus-based universities teach programming in laboratory classes in which students can learn to program solo, or in variously-sized groups. Many institutions also teach pair programming: two programmers work "side-by-side" at a computer when programming to solve a problem. The person typing the code – the driver – and the person watching that coding to spot any errors or ways to improve the code – the navigator – switch roles regularly. Both partners in the pair voice their thinking, sharing their questions and exchanging ideas. Previous research from UK, Europe and USA indicates that pair programming can lead to improved quality of programming, enhance programming skill and increase self-confidence when programming. This work is to investigate if such benefits also can accrue for distance learners pair programming with a remote partner online. Our motivation stems from the reality that in our university, the majority of students studying to learn how to program do so individually and remotely via distance learning. Therefore, whilst we acknowledge the work of many educators in exploring the benefits to students of pair programming, we are acutely aware that the bulk of that work is set in the context of face-to-face classes. The work described here is from a pilot study to investigate if Remote Pair Programming can bring similar employability, social and community benefits to students and thereby improve their learning experience. (Colleagues are researching the technical aspects of Remote Pair Programming and will report their findings in due course.)

Our research is designed to identify recommendations for educators incorporating pair programming into distance learning. Those techniques and approaches found to be feasible, manageable and beneficial will be recommended as guidelines for embedding pair programming in undergraduate and postgraduate modules that teach programming. It is anticipated that positive outcomes might relate to student

[a] https://orcid.org/0000-0002-6813-9020
[b] https://orcid.org/0000-0002-0269-8284

satisfaction, student confidence and self-esteem, improved student employability and improved module feedback ratings for those modules. This paper describes the results from a pilot study designed to confirm the feasibility and efficacy of the research and to inform its methodology when scaled.

2 RELATED WORK

Computing educators have been researching pair programming for 20 years, spurred on by the innovations of Williams at North Carolina State University.

2.1 Benefits of Pair Programming

As early as in 2001, Williams and Kessler found that pair programming allowed students "to learn new languages faster and better than with solitary learning" (Williams and Kessler, 2001). Consistent findings have included that student pair programming has several benefits, not only to the quality of their work products e.g. (Hanks et al., 2004), (Smith et al., 2017) but also with respect to student satisfaction, e.g. (Williams and Upchurch, 2001), (Hanks et al., 2004) and confidence, e.g. (Zacharis, 2011). Considering CS1, (Wood et al., 2013) found that pair programming led to a generally increased sense of community among the students. However, the title of the Williams and Kessler work highlights the issue for the current authors: "Experiments with industry's "pair-programming" model *in the computer science classroom*" (our emphasis) (Williams and Kessler, 2001).

2.2 Remote Pair Programming

Numerous researchers have published supportive findings but most often in the context of classrooom-based teaching and learning in campus based universities and colleges. Those familiar with the self-explaining work of Chi from the 1990s, e.g. (Chi et al., 1994), will appreciate some of the reasons that students are likely to benefit from pair programming - but we question why the benefits ascribed to campus-based circumstances cannot also accrue to students of distance learning. If students in classrooms who practise pair programming are more self-sufficient and demonstrate higher-order thinking skills (Williams et al., 2002), the question arises if the same benefits of pair programming - which go beyond programming skill - are available to distance learning students and how can that be managed? Some early work suggests that distributed pairs obtain many of the same benefits as co-located pairs, including the fostering of teamwork, communication and confidence e.g. (Baheti et al., 2002), (Stotts et al., 2003), (Nagappan et al., 2003), (Hanks, 2005). However a 2010 comparison of distributed and co-located pair programming found that whilst both approaches led to positive experiences for students, the former resulted in slightly less satisfaction than the latter (Edwards et al., 2010). More recent work suggests the promise of students generating their own pair programming events in a large MOOC (McKinsey et al., 2014).

2.3 Guidelines

In 2008, Williams et al. derived a set of nine classroom management guidelines "for successfully implementing pair programming in the classroom", adding a further two in the context of an HCI course. As well as providing insights for educators wishing to adopt pair programming in face-to-face teaching circumstances, these authors also recognised the "need to coordinate schedules when pair programming is required outside of a classroom or laboratory setting" (Williams et al., 2008). Here, we describe a pilot that is part of work to identify guidelines for tutors and students to support successful pair programming in an online learning environment. The remainder of the paper is structured as follows: section 3 describes the method adopted for the pilot study; section 4 presents findings from the students' and tutors' feedback; section 5 summarises the lessons learned; section 6 concludes the paper with a summary of future work.

3 METHOD

Our approach was designed to consider three techniques of experiencing Remote Pair Programming to identify any perceived benefits. The three methods were:

1. Passive: watching a video recording of two expert tutors pair programming "side-by-side"
2. Indirect Participation: watching two tutors pair programming live online, with the opportunity to interact with them either during or at the finish
3. Direct Participation: working on a pair programming task with a (remote) student partner online.

Five undergraduate student volunteers participated in the pilot study and were surveyed, online, for their perceptions of employability-related skills following their participation at each phase of the pilot. Ten survey statements related to how students felt:

- I am able to work well with others, communicate orally and give / receive feedback

- I am able to analyse facts and circumstances and ask the right questions to diagnose problems.
- I feel connected to others in this course.
- I trust others in this course.
- I am able to communicate orally in a clear and sensitive manner which is appropriately varied according to different audiences.
- I am able to take initiative and action unprompted to achieve agreed goal.
- I am able to deal confidently with challenges.
- I am able to reflect on my own practice and strengths and weaknesses.
- I feel that other students help me learn.
- I am able to analyse, reason and problem solve.

Following each statement were three possible responses: more than before, no change, and less than before. The survey statements were derived from two questionnaires (Rovai, 2002) and (Jackson and Chapman, 2012) and our institution's Employability Framework. Drawing on each of these sources, the set of ten statements was framed to draw comments about self-efficacy, team working, communication skills, collaboration, self-awareness, connectedness and problem-solving ability. An answer style was developed to allow for speedy online responses (select a radio button per question). Survey responses were anonymous. The students were asked to mark which of the responses best reflected their feelings about the statement, and to provide any further comments via a text box after each statement. They were asked to complete the survey after experiencing each of the three methods described above.

Ethical permission was obtained from the university's Student Research Project Panel and from the Module Team Chair of the module being studied (a CS1 Introduction to computing and information technology module that includes programming using Python as well as patterns and algorithms). Two experienced tutors (authors 3 and 4 of this paper) in the research team performed the first phases.

3.1 Phase 1

A programming task of appropriate academic level was set by one of the tutors in the research team. It was designed to take approximately 30 minutes to complete, using patterns and algorithms already introduced to the students as part of their module studies. Pair programming guidelines (Zarb and Hughes, 2015) were issued to both the tutors in advance. Video recordings were made of the two module tutors pair programming at the same computer in a laboratory, for participants to watch (repeatedly if desired) to gain insights into how to pair program. Two recordings were made (front view to show the tutors' faces as they programmed and rear view to show the screen as well as who was driving and who was navigating at any point in time). Tutors used an adjacent whiteboard whilst planning and/or designing their solution. Screen capture was also made to allow details of the coding to be viewed clearly. Video editing software was used to display three views, such as tutors, code, and whiteboard or programming task (Figure 1).

Figure 1: Video of tutors, code and programming task.

3.2 Phase 2

Four participants watched live streaming of the two module tutors pair programming online using Adobe Connect for the connection and Python with the programming environment IDLE. (One of the original participants had to leave the study prematurely because of personal work commitments.) Whilst other more sophisticated pair programming tools exist, such as USE Together (https://www.use-together.com/), the choice of Adobe Connect was a pragmatic one: the university has licensed this software for use with all students and staff, and so both groups had experience of the software. Furthermore, given that a single module may have over 2,000 students, the licensing costs of other solutions were judged to be worthy of investigation *only if/when* the benefits of Remote Pair Programming had been established.

A second programming task, set by the same tutor, was used and again designed to be completed within approximately 30 minutes. Both during and after observing the pair programming in real time, the students were able to comment or ask questions of the tutor pair, either directly using their microphones or via the software's chatbox (Figure 2). Each tutor's webcam was switched off after some minutes to minimise any issues relating to bandwidth.

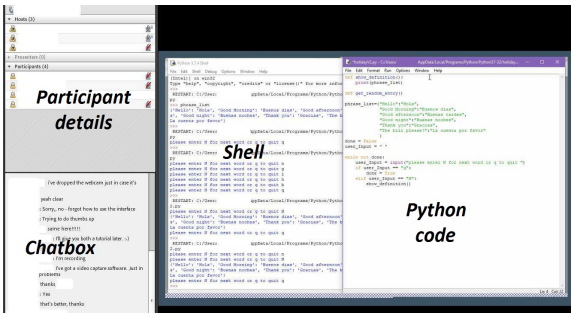

Figure 2: Live streaming of pair programming.

3.3 Phase 3

Four participants conducted Remote Pair Programming themselves using Adobe Connect and Python, with IDLE, in the same way that they had observed the tutors pair programming. These four participants were paired by the research team to ensure an appropriate match of perceived confidence and experience, according to their self-assessment. Each event was scheduled to suit the participants' circumstances, e.g. day of the week and time of the day. Pair programming guidelines (Zarb and Hughes, 2015) were issued to both pairs in advance. A third task, set by the same tutor, was designed to be completed in no more than an hour. A member of the research team advised the participants during the initial few minutes of their partnering and pairing online, to ensure that they were able to use both Adobe Connect and Python, switching between driving and navigating. Once they had established successful communication and programming, they were left to complete the task themselves without further monitoring.

4 INITIAL FINDINGS

Our initial findings are based upon the participants' survey responses, comments from tutors, the Module Team Chair, colleagues who also participate in and teach about pair programming, and the observations of the research team themselves.

4.1 Filming of Tutors

Survey responses provided insights into the relative value of watching the pair programming video. There were some positive comments relating to awareness-raising: it was "a valuable exercise" and that "watching the tutors interact was educational". However, whilst none of the survey responses was negative, two-thirds of the survey responses indicated that watching the video had made no difference to participants' feelings about participating in pair programming, which is unsurprising. In summary, the pilot study established that the video method absorbed the greatest time in preparation and post-filming effort but attracted the smallest set of positive comments. For instance, the inclusion of three different video aspect views was judged by tutors to be important so that participants could focus upon their preferred aspect at any stage without the need to keep replaying the video. However this necessitated the use of a combination of cameras and screen capture facility, and considerable subsequent editing.

4.2 Tutors Pair Programming Live

Feedback about the live streaming event was more positive, not least in terms of developing a peer community. Two-thirds of the survey responses indicated that watching the tutors perform pair programming live led to increases in nine of the ten question areas. The exception was "I feel I can take initiative and action unprompted to achieve agreed goals.", which attracted a negative response around having to explain actions and keep synchronised with the partner.

No students voiced any questions during the event. For greater interaction during this approach, it seems that additional preparation is needed to increase students' confidence to interrupt and question the tutor pair whilst they are programming. As one student noted: "As it was the first time participants were hesitant to speak I think, I'm sure this will be different going forward." The chatbox was mostly used for (apparently) inconsequential comments. In reality, those social interactions were evidently important as introductions and ice-breakers, student-to-student.

4.3 Students Pair Programming Live

Feedback from the two sets of pair programmers provided valuable information relating to the benefits and the difficulties of live pair-programming with a peer that they did not know. As with watching tutors pair programming live, two-thirds of the survey responses led to increases in nine of the ten question areas, the exception again being the statement about taking initiative and action unprompted, which was negative for the same reason. Positively, the experience led to increased confidence with reflection on personal strengths and weaknesses, e.g. "it was good to see the strengths and weaknesses I have, mainly in how I communicate, not so much with the programming itself." Other positives related to reduction in isolation: "It was good to meet others studying, with the

OU you can forget there are others out there doing this with you" and confidence with communication: "Despite differing backgrounds and no previous communication, the partnering went well".

Turn-taking was one area of concern, however. Evidently students needed more guidance about this aspect of working with a remote partner:

> "I felt initiative and unprompted action were more difficult when programming with another student as we had to keep our ideas in sync with each other which meant I felt every thought process had to be explained fully and "confirmed" by the other person before it was tried."

Also evident was the need to pair students according to an appropriate level of confidence or experience:

> "I once again found that I sometimes have difficulty keeping up with the thought process of someone else who may have already figured out the next step in the problem before I have"

Using Adobe Connect without video (for bandwidth reasons) led to some frustrations around the absence of visual cues:

> "I found the programming with another student a bit challenging. This was partly down to how, as I couldn't see the other person, I didn't know when they were going to speak, so it led to times where we would accidentally talk over each other."

4.4 Student Experience

The survey responses demonstrated that all of the methods used led to *some* feelings of increase in team working, problem solving confidence, sensitive communication manner, taking initiative, and self-assessment. Examples of explanatory free text comments included:

> "I think this is good experience for communicating with different audiences and seeing where my strengths and weaknesses are with how I communicate."

> "I definitely feel that this improved my feeling of participation within my distance learning course. Sometimes it's easy to feel as if tutors and other students are very remote."

Although there was very limited interaction with the tutors during the live streaming event, the students made positive observations, such as:

> "The chat between the two tutors was very courteous. Even when one of them made an obvious mistake or just got lost in the particular language the error was pointed out in a gentle way."

Even watching the video of tutors pair programming was judged by some to be of benefit:

> "I picked up some tips from watching the video, such as how the problem statement was approached (one person read and the other listened) and how the problem was tackled."

> "I think that from watching the 2 tutors communicate and coordinate the work, it's given me something on which to base how I work with someone else"

5 EXPERIENCE GAINED

There were a number of limitations in the pilot study reported here: in particular, the sample size was very small and the students were all volunteers and possibly predisposed to view pair programming experiences positively. Nonetheless the experience of this pilot study corresponds to various publications related to co-located pair programming research, particularly with respect to the challenges, e.g. as summarised by (Hanks et al., 2011) with respect to pairing and guiding students. We offer here some of the lessons learned from this work that will be applied in our forthcoming scaled-up project, in the belief that these will be of value to educators considering embedding pair programming into distance learning courses.

5.1 Technologies

One simple lesson learned for video capture related to the differing frame rates of different cameras and screen capture facility: having omitted to synchronise these before filming, considerable effort was needed to stitch together the various components seamlessly.

More importantly, it appears that is not *essential* to use expensive commercial professional pair programming tools to benefit the student experience. Adobe Connect was adequate for students to interact, to develop a sense of community, to reflect upon their own communication skills and to develop their team working experiences. This is similar to one of the findings of (Stotts et al., 2003) that effective software development is feasible with a few simple and widely-available tools. Nonetheless, (Edwards et al., 2010) speculated that whilst generic collaboration software available to universities provided a viable method of distributed pair programming, it could cause to frustrations for user-based and technological reasons.

Various researchers continue in their work to develop or investigate feature-rich collaborative tools to support pair programming at a distance, e.g. (McKinsey, 2015). (Urai et al., 2015) concluded that distributed pair-programming systems should have functions to enable easy communication and partner change. The advent of MOOCs has further stimulated this area of work, e.g. (Ghorashi and Jensen, 2016), (Ghorashi and Jensen, 2017) and (Staubitz and Meinel, 2018).

5.2 Arranging Pairs

A number of factors are important when pairing students to work together. Areas listed here were those most noticeably important to our student participants.

Some students reported self-consciousness about being sufficiently capable ("I am a very slow learner ..."; "sometimes you do get stumped"). Some asked, unprompted, about being paired with a person at the same level as themselves. This corresponds to early findings of the analysis of pair programming in a classroom environment (Williams et al., 2006), that students are compatible with partners whom they perceive of similar technical competence, and later work e.g. (Bowman et al., 2019) describing the negative effects of being paired with a more experienced programmer. An obvious difficulty in distance learning is for students to identify others with similar skill levels. Recent initiatives e.g. (Deeb et al., 2018), (Berland et al., 2015) are investigating the use of analytics to shape the formation of partnerships and groups.

Other obvious areas relate to the nature of distance learning itself: students may be in different time zones, or working full-time and not able to partner during the day, or working in the evenings or weekends and prefer to schedule their study at unusual hours of the day. Unlike in a campus-based situation, there are no fixed class times to enforce pairings. When arranging pairs, a further factor to consider is the stated motivation of the individuals. In the work reported here, these ranged from improving communication skills to improving programming skills and reducing isolation. It may be that students will particularly benefit from being paired with similarly-motivated individuals.

A follow-up pilot study is now beginning. We have enquired of students who have submitted consent forms to participate if they had a preference to pair with a person of the same gender, different gender or if that did not matter. Interestingly, all 25 who have responded to date stated that the gender of the partner did not matter. This presents a challenge for the researchers, given the findings of (Katira et al., 2005) that pairs with different gender are less likely to report compatibility than pairs of the same gender. A related issue was identified in recent systematic literature review of distributed pair programming: (da Silva Estácio and Prikladnicki, 2015) were unable to identify any study relating to cultural aspects in practice or in teaching perspectives.

Performing a thematic analysis of CS1 students' reflections on pair programming, (Celepkolu and Boyer, 2018) found pair programming to be motivating and that it contributed to social growth – but high-achieving students were less positive and recognised fewer benefits than other students. Muller and Padberg investigated empirically what they called the *feelgood factor* in pair programming: their primary guideline was "First of all, make your pairs feel good!" (Muller and Padberg, 2004). Our work suggests that their recommendation is as important for distance learning as it is in the classroom.

5.3 Guiding Students

Distance learning tutors are experienced users of systems such as Adobe Connect for giving online tutorials and individual support sessions for students. Tutors know the difficulties of communications timelags, of the need to "give space" for others to speak, of the awkwardness that results from interrupting, and of the need to encourage interaction. This work demonstrated that students do not have that experience – they need to be provided with guidelines about turn-taking and verbalising, and more practical rehearsal time if they are to overcome the difficulty of interacting with a remote partner without visual cues that indicate when the other person is about to speak. Nagappan et al. recognised this early on:

> "Distributed pair programmers absolutely must be willing to speak while they work. They must explain what they are doing as they are doing it or the navigator quickly gets lost. This is so essential that programmers who are not willing to speak almost continuously should probably not try to work this way." (Nagappan et al., 2003)

Initially, it was believed that a ten-minute rehearsal with the communication between partners was sufficient for each of the students to take ownership of the pair working without further tutor intervention. However the feedback from students subsequently made clear that further guidance was required. The guidelines used in this study (Zarb and Hughes, 2015) do not address the absence of cues in remote working; we plan to update these guidelines to accommodate distance learners. The first guideline identified by (Williams et al., 2008) holds even more true for

Remote Pair Programming students: "Students need training in pair programming in a supervised setting to experience the mechanics of successful pairing". It appears that such guidelines may also be needed for the implementation of distributed pair programming in industry (da Silva Estácio and Prikladnicki, 2015).

5.4 Setting Appropriate Tasks

Setting an appropriate level of difficulty is important: if the task is too easy, the tutors' progress appears unnaturally slow and artificial; if too challenging, students cannot discern how decisions are taken unless the tutors are particularly good at verbalising. One reported benefit of the video and the live streaming was when students witnessed the tutors hesitating and making mistakes. Both the video and the live event were engaging precisely because the tutors were natural and their dialogue was unscripted: the students witnessed mistakes occurring and the recovery process. Onlookers to those aspects of programming certainly recognised that the experience was authentic.

5.5 Additional Needs

It is essential that students with special needs are considered: pair programming should be possible for *all* our students. Students might not disclose all the factors (e.g. autism, visual impairment) that tutors should understand when pairing people or selecting technologies.

6 CONCLUSIONS

Pair programming means two people programming software together. In classroom circumstances, it brings benefits to students including greater sharing of expertise and fewer errors. Whilst routinely used in industry and taught in face-to-face university programming lab classes, its use in distance learning is less well understood. The value of successful pair programming in a distance learning context goes beyond the development of programming expertise: it may reduce any feelings of isolation resulting from having fewer networking opportunities than campus-based students. In this pilot study, Remote Pair Programming was associated positively with feelings about communication skills and team-working, which may enhance employability. For a successful distance learning pair programming experience, educators need awareness of the standard issues that researchers have published for face-to-face pair programming (e.g. the importance of *perception* of ability with respect to allocation of partners) AND additional issues such as turn-taking advice, scheduling constraints, and ensuring that relevant facilities are in place for any additional special needs. Put simply, planning for the human and social interaction aspects may be more important than planning to use the most contemporary technology. As noted in the Human-computer interaction paper 'Distance Matters':

> "Collaborative work at a distance will be difficult to do for a long time, if not forever." (Olson and Olson, 2000)

Clearly this preliminary work needs to be repeated at scale. Our future work includes that extended research, and further investigation of the challenges of pairing distance learning students who do not know any other students in their class. Some participants also will be invited to a focus group and others interviewed to further probe the employability and community benefits identified and the student satisfaction. We plan to review the original guidelines of (Williams et al., 2008) to give recommendations and produce a revised version appropriate to 2020 distance learning.

ACKNOWLEDGEMENTS

Our thanks are due to students and colleagues in The OU, to Quality Enhancement Themes (Scotland) for support of the pilot, and to The OU eSTEeM team for supporting the follow-up project. Thanks are also due to the anonymous reviewers who provided valuable comments upon an earlier version of this paper.

REFERENCES

Baheti, P., Williams, L., Gehringer, E., and Stotts, D. (2002). Exploring pair programming in distributed object-oriented team projects. In *Educator's Workshop, OOPSLA*, pages 4–8.

Berland, M., Davis, D., and Smith, C. P. (2015). Amoeba: Designing for collaboration in computer science classrooms through live learning analytics. *International Journal of Computer-Supported Collaborative Learning*, 10(4):425–447.

Bowman, N. A., Jarratt, L., Culver, K., and Segre, A. M. (2019). How prior programming experience affects students' pair programming experiences and outcomes. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, pages 170–175.

Celepkolu, M. and Boyer, K. E. (2018). Thematic analysis of students' reflections on pair programming in cs1. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pages 771–776.

Chi, M. T., De Leeuw, N., Chiu, M.-H., and LaVancher, C. (1994). Eliciting self-explanations improves understanding. *Cognitive science*, 18(3):439–477.

da Silva Estácio, B. J. and Prikladnicki, R. (2015). Distributed pair programming: A systematic literature review. *Information and Software Technology*, 63:1–10.

Deeb, F. A., DiLillo, A., and Hickey, T. J. (2018). Using fine grained programming error data to enhance cs1 pedagogy. In *CSEDU (1)*, pages 28–37.

Edwards, R. L., Stewart, J. K., and Ferati, M. (2010). Assessing the effectiveness of distributed pair programming for an online informatics curriculum. *ACM inroads*, 1(1):48–54.

Ghorashi, S. and Jensen, C. (2016). Supporting learners in online courses through pair programming and live coding. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 738–747. IEEE.

Ghorashi, S. and Jensen, C. (2017). Integrating collaborative and live coding for distance education. *Computer*, 50(5):27–35.

Hanks, B. (2005). Student performance in cs1 with distributed pair programming. *SIGCSE Bull.*, 37(3):316–320.

Hanks, B., Fitzgerald, S., McCauley, R., Murphy, L., and Zander, C. (2011). Pair programming in education: A literature review. *Computer Science Education*, 21(2):135–173.

Hanks, B., McDowell, C., Draper, D., and 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*, ITiCSE '04, page 176–180, New York, NY, USA. ACM.

Jackson, D. and Chapman, E. (2012). Non-technical competencies in undergraduate business degree programs: Australian and uk perspectives. *Studies in Higher Education*, 37(5):541–567.

Katira, N., Williams, L., and Osborne, J. (2005). Towards increasing the compatibility of student pair programmers. In *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.*, pages 625–626. IEEE.

McKinsey, J. (2015). Remote pair programming in a visual programming language. *Technical Report No. UCB/EECS-2015-139*.

McKinsey, J., Joseph, S., Fox, A., and Garcia, D. D. (2014). Remote pair programming (rpp) in massively open online courses (moocs). In *Proceedings of the 2014 conference on Innovation & technology in computer science education*, pages 340–340.

Muller, M. M. and Padberg, F. (2004). An empirical study about the feelgood factor in pair programming. In *10th International Symposium on Software Metrics, 2004. Proceedings.*, pages 151–158. IEEE.

Nagappan, N., Baheti, P., Williams, L. A., Gehringer, E. F., and Stotts, D. (2003). Virtual collaboration through distributed pair programming. Technical report, North Carolina State University. Dept. of Computer Science.

Olson, G. M. and Olson, J. S. (2000). Distance matters. *Human–computer interaction*, 15(2-3):139–178.

Rovai, A. P. (2002). Development of an instrument to measure classroom community. *The Internet and higher education*, 5(3):197–211.

Smith, M. O., Giugliano, A., and DeOrio, A. (2017). Long term effects of pair programming. *IEEE Transactions on Education*, 61(3):187–194.

Staubitz, T. and Meinel, C. (2018). Collaborative learning in moocs approaches and experiments. In *2018 IEEE Frontiers in Education Conference (FIE)*, pages 1–9. IEEE.

Stotts, D., Williams, L., Nagappan, N., Baheti, P., Jen, D., and Jackson, A. (2003). Virtual teaming: Experiments and experiences with distributed pair programming. In *Conference on Extreme Programming and Agile Methods*, pages 129–141. Springer.

THE (2020). Times higher education world university rankings https://www.timeshighereducation.com/world-university-rankings/open-university.

Urai, T., Umezawa, T., and Osawa, N. (2015). Enhancements to support functions of distributed pair programming based on action analysis. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, pages 177–182.

Williams, L., Layman, L., Osborne, J., and Katira, N. (2006). Examining the compatibility of student pair programmers. In *AGILE 2006 (AGILE'06)*, pages 10–pp. IEEE.

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

Williams, L. and Upchurch, R. L. (2001). In support of student pair-programming. *SIGCSE Bull.*, 33(1):327–331.

Williams, L., Wiebe, E., Yang, K., Ferzli, M., and Miller, C. (2002). In support of pair programming in the introductory computer science course. *Computer Science Education*, 12(3):197–212.

Williams, L. A. and Kessler, R. R. (2001). Experiments with industry's "pair-programming" model in the computer science classroom. *Computer Science Education*, 11(1):7–20.

Wood, K., Parsons, D., Gasson, J., and Haden, P. (2013). It's never too early: pair programming in cs1. In *Proceedings of the Fifteenth Australasian Computing Education Conference-Volume 136*, pages 13–21.

Zacharis, N. Z. (2011). Measuring the effects of virtual pair programming in an introductory programming java course. *IEEE Transactions on Education*, 54(1):168–170.

Zarb, M. and Hughes, J. (2015). Breaking the communication barrier: guidelines to aid communication within pair programming. *Computer science education*, 25(2):120–151.