Investigating the Perceived Benefits to Computing Students of Remote Pair Programming

Keywords: pair programming; distance learning; community

Janet Hughes, Ann Walshe, Bobby Law, Brendan Murphy

08.10.2021

<u>Executive Summary</u>

Pair programming is typically taught in face-to-face classes to enhance students' coding expertise. Distance learning students have little or less opportunity for that enhancement. Difficulties in providing a pair programming experience for students are not exclusive to the OU: they were experienced by other higher education institutions which switched delivery modes as a result of the 2020-021 pandemic, being forced to design, develop and implement online and distance learning education. Whilst the technical logistics of remote pair programming are not particularly problematic, the design of the educational logistics needs sensitivity. Educators may wonder if the effort required to arrange remote pair programming is worthwhile, particularly given that partnerships need to be arranged between students who have never met, as is the situation for the OU.

This remote pair programming project was designed to explore the non-technical benefits of different methods of experiencing remote pair programming. Results indicate that students perceived that working remotely with another student increased their verbal communication skills and their ability to collaborate, problem solve, make decisions, and take initiatives –- and slightly enhanced their sense of learning from others. Paradoxically, the area which most merits further support is in feeling connected to others in a module, which is a key issue for Open University students and for others in education in a locked-down pandemic world. However, other benefits identified by participants related to reassurance, mentoring, and learning to ask for help –- and student participants were keen for further understanding of the experience of programming in the real world. We provide recommendations for module teams and tutors to consider when designing remote pair programming experiences.

Aims and scope of the project

Our computing students learn how to program largely individually and remotely. As reviewed in Zarb & Hughes (2015), previous research indicates that pair programming can lead to improved quality of programming, enhance programming skills and increase self-confidence when programming. This project aimed to investigate the benefits to our students of engaging in remote pair programming in their learning. In particular, the investigation aimed to go beyond academic learning to explore community and employability benefits, both of which are relevant to NSS amongst other measures of student satisfaction.

This work is designed to test if social and community experiences, as identified by related research in industry and in campus-based classrooms, can be gained by distance learners of programming in higher education whose pair partners are online rather than sitting alongside them in a laboratory.

Activities

Meta-studies, such as that of Umapathy (2017) and Hannay (2009), have examined a number of quantitative studies, concluding that pair programming brings value to academic performance and programming quality. In this work, qualitative research was the primary approach used in order to discover students' perceptions of the non-technical benefits of remote pair programming. A number of published studies confirm that a qualitative research methodology is valuable to gain insight into situations of pair programming, e.g. Lewis (2015), Ying (2019).

Study design
The research was designed to be descriptive, to allow participants to identify their feelings about three different types of experience in the specific situation of one of their preliminary programming modules. Textual data would be gathered by survey research and focus group to gain insights into the feelings of a small sample of distance learning students. Data collection would be completed over a short period during their module studies and without an undue pressure on their time. All aspects of the work was formative and without judgment or impact upon assessments. Ethical agreement for the study was obtained in advance of students being invited to participate.

Participants
Volunteer undergraduates were sought from module TM112. Lasting 21 weeks, about one third of the module introduces Python programming. All students contacted had earlier completed module TM111, which introduced programming using a block-based version of Scratch rather than a text-based language. By email, students on the Python module were provided with a summary of the research project and invited to participate. No condition was set upon participation, for example relating to experience, race, gender, age or occupation, and no demographic data was collected from the students. It was made clear that participation was unrelated to the module assessments or results. Having submitted consent forms, 43 students participated in the project (approximately 17% of the students registered). Broadly in keeping with the gender balance for the module, 25% of the participants were female.

Procedure
OU students are often provided with videos about new topics to support their learning, or are offered the opportunity to watch tutors performing in a live demonstration of new procedures or techniques. Each of these may provide students with a vicarious experience of pair programming. In this project, three methods of experiencing pair programming therefore were offered, ranging from fully vicarious to fully direct. All students who had confirmed their consent were invited to participate via all three methods:

A. passive participation (watching a video of two tutors pair programming)
B. indirect participation (watching two tutors remote pair programming in real time)

C. direct participation with an online student partner (remote pair programming).

In each circumstance, the task to be solved had been devised by the tutors to correspond to the level of difficulty and learning topics the participants were studying at that stage in their module.

Passive participation
In method A, a video of two tutors pair programming at a computer in a laboratory situation was made available for participants to watch at a time or times of their convenience. The video from method A could be watched before or after methods B and C, or before **and** after the other methods. Audio accessibility concerns were addressed by providing a transcript on request; one participant made this request. Pair programming guidelines (Zarb, 2015) had been issued to the tutors in advance. Python was used by the tutors to solve a modest task, lasting less than an hour.

Indirect participation
In method B, the two tutors demonstrated further pair programming in real time using Adobe Connect (which was the small pilot phase 1) and Microsoft Teams (which was the larger phase 2 study). Again, Python was used by the tutors to solve a different but equally modest task, lasting less than an hour. In phase 1, tutors used a regional Adobe Connect room for the work. A guidance sheet was developed to outline how to share a programming screen using Adobe Connect. Microphone and webcam in Adobe Connect were used in the normal way. In phase 2, tutors and students were all enrolled in a Microsoft Team, with all participants also enabled via Microsoft Teams to speak and share video, and use chat. Tutors also were given the capability to share their screen or desktop, which is the method by which they switched roles from driver to navigator: one tutor began as the driver, launching the Python IDLE, positioning a shell window and an editor window side-by-side, and saving the work to his PC; that tutor shared his desktop with the navigator tutor. When it was time to hand over the driver role, the driver tutor gave control to the navigator tutor (hovering the mouse at the top of the screen and choosing `Give control` to the named tutor from the pop-up toolbar). A guidance sheet was developed to outline how to do this. If the navigator wanted to take over as driver, the same pop-up toolbar was used to request control.

Only method B had a fixed date and time, since it was necessary for the tutors to agree an appropriate time for their real-time remote programming; participants were notified of that date and time and invited to attend. Microsoft Teams allowed for the tutors faces to be visible whilst they discussed the given task, planned and then attempted to reach a solution, with screen sharing and swapping of control as they progressed their solution. Participants were able to interact with each other using audio and video at the start and again at the end, and ask questions or make observations to the tutors at any time by means of the chat. Audio accessibility concerns were addressed by offering Microsoft Teams live captions.

Direct participation
For method C, participants were asked about any preferences for partnering with another student to solve a small Python task anticipated to last no more than one hour. In particular, participants were asked for any preferences for day of the week, time of the day, gender of the partner, and if there was anything else they wished to disclose before pairs were arranged. One participant had identified a preference to work with another female. One participant expressed a concern about their level of English language. A small number of participants expressed some anxiety about their novice level of programming ability.

Participants were then partnered to ensure close matches of programming experience (as judged by their performance in the previous block-based Scratch module) and confidence (which was the aspect most commented upon in response to preferences), as per the published literature of best practice in pair programming. Scheduling needs were considered next to match participants' availability, and then any other concerns disclosed. All sessions were scheduled to take place after method B had concluded, to avoid a clash of the pair work with a major assignment submission date. None of the pairs had worked with each other before or met each other, either face-to-face or online.

Pair programming guidelines were issued to the students in advance of the task. In phase 2, when Microsoft Teams was used, each pair was given a private channel for their work and reassured that no recording was being made of their actions or discussion. Guidance was sent to each pair about the use of Microsoft Teams to give and take control for driving and navigating while pair programming (using the same techniques described in method B). A member of the project team accompanied each pair at the start of their pair programming session to introduce the students to each other and encourage some ice-breaking conversations, until it was clear that a pair was able confidently to communicate and share Python. At that point, the research team member left Microsoft Teams, allowing each pair privacy to tackle a given Python problem of a similar level of difficulty earlier tackled by tutors.

Data Collection

Survey data
After experiencing each technique, students were asked to complete an online survey to describe their perceptions of the benefits of pair programming to social and community feelings, including giving comments. An independent student volunteer had earlier provided feedback on the ten survey items, one of which was amended considerably as result of the pilot phase of the research. Each survey item asked students which of five responses -- strongly agree, agree, neutral, disagree, and strongly disagree -- best reflected their own feelings about a given statement after participating in one of the three pair programming methods. The ten survey items were:

1. I feel that I can work with others using collaborative communication, problem solving, discussion and planning.
2. I feel that I can analyse facts and circumstances and ask the right questions to diagnose problems.
3. I feel connected to others in this course.
4. I feel I can make appropriate decisions, in light of available information, in sensitive and complex situations.
5. I feel that I can communicate orally in a clear and sensitive manner which is appropriately varied according to different audiences.
6. I feel that I can take initiative and action unprompted to achieve agreed goals.
7. I feel that I can deal confidently with challenges.
8. I feel that I can reflect on my own practice and strengths and weaknesses.
9. I feel that other students help me learn.
10. I feel that I can analyse, reason and problem solve.

Items 3 and 9 were based on Rovai's Classroom Community Scale (Rovai, 2002), which was described as *"a test instrument that can assist educational researchers in studying community in virtual classrooms and help identify course design and instructional delivery that best promotes the development of* 'community'".

Item 3 focuses on the feelings of connectedness with other community members, whereas item 9 focuses on the feelings of interaction with others for learning, for example by sharing understanding.
Items 1 (collaboration), 2 (diagnostic problem solving), 4 (decision-making), 5 (verbal communication), 6 (initiative), 7 (self-efficacy), 8 (meta-cognition), and 10 (analytical problem solving) were derived from Jackson and Chapman's identification of behaviours associated with generic non-technical competencies most relevant to industry, particularly in the field of business (Jackson, 2012).

Each of these ten survey items were checked to be relevant to elements of the OU's employability framework.

Focus group data
Reviewing the survey data results and comments, we identified topic areas to be probed further by means of a focus group. Topics were discussed, reviewed, and five key areas identified as starter questions with up to three associated prompts for the focus group facilitator to use if necessary. Questions were reviewed before use by an independent student to confirm their clarity, open nature, and that there was a logical sequence from general to specific. Microsoft Teams was the channel for the discussions.

Four participants had volunteered to participate in the focus group. Subsequently one was unable to participate and therefore three students met once using Microsoft Teams to discuss the five topic areas. Participants were given prompts for the discussion areas in advance. The focus group event itself was conducted by another independent student with appropriate experience, and recorded then transcribed for later analysis.

Data analysis

Quantitative survey data was analysed by determining the medians for the Likert-type data and comparing the frequencies of different categories for each method of participation. ANOVA and Tukey's HSD tests were run to determine any statistically significant differences in experiences using the different methods. Despite only two categories *not* suggesting a statistically significant difference, these results are not reported here since the collected data was Likert-type, and therefore it was not possible to verify that the participants viewed the categories as being on an interval scale. Instead, median values were determined.

Free text comments from the survey and the descriptive statistics were used to identify themes for exploration using the focus group. Two members of the project team independently reviewed the focus group recording and transcription, and coded key ideas. A third member of the project team reviewed these, and thereafter the main themes were confirmed.

Findings

1. Survey
Results for all survey items for each method of participation are presented in Figures 1-3. Note that on two occasions in the passive participation survey (Figure 1), and one further occasion in the direct participation survey (Figure 3), one student did not select a response for a survey item. Survey items, as listed in the Data Collection section above, are labelled 1-10 on the vertical axis.

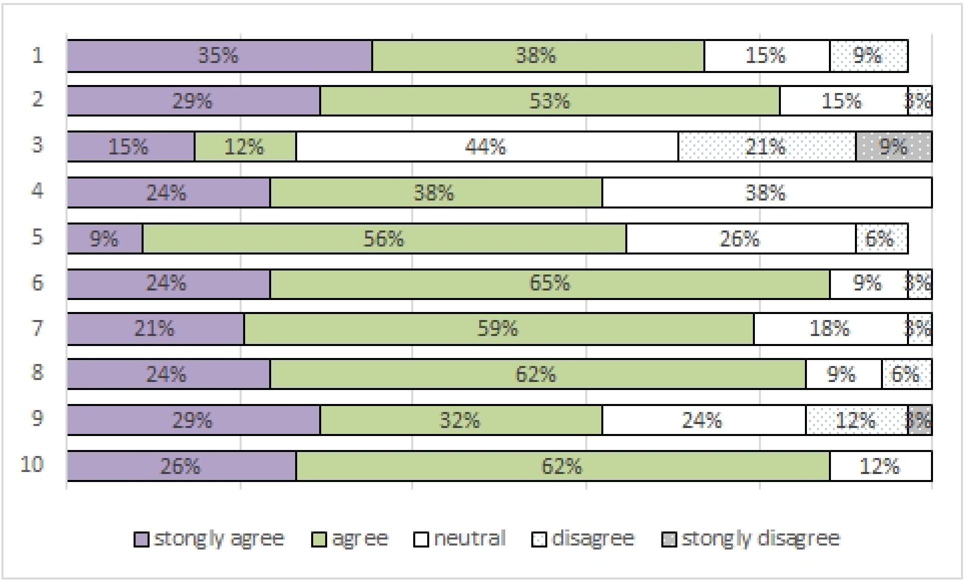


Figure 1: Passive participation (video) N=34

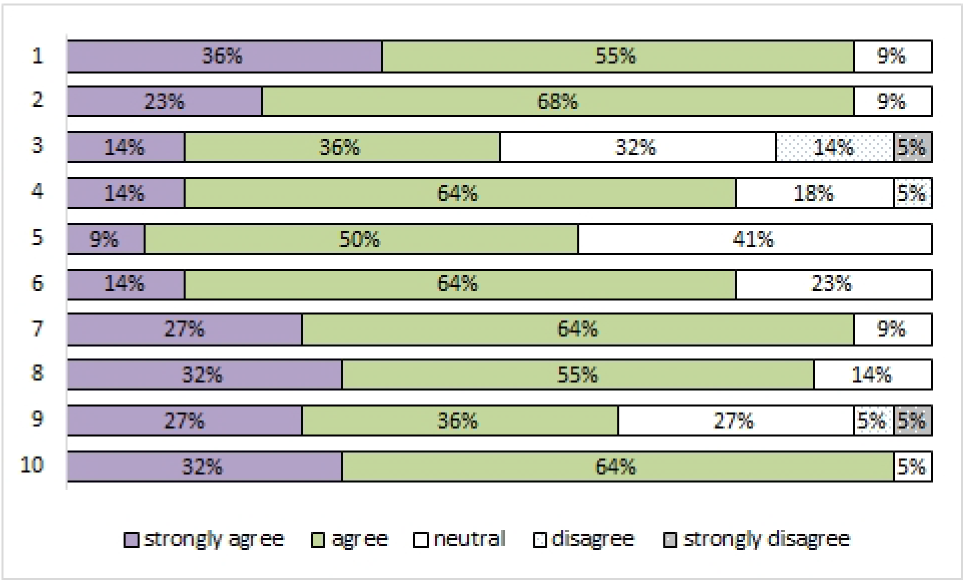


Figure 2: Indirect participation (live tutoring) N=22

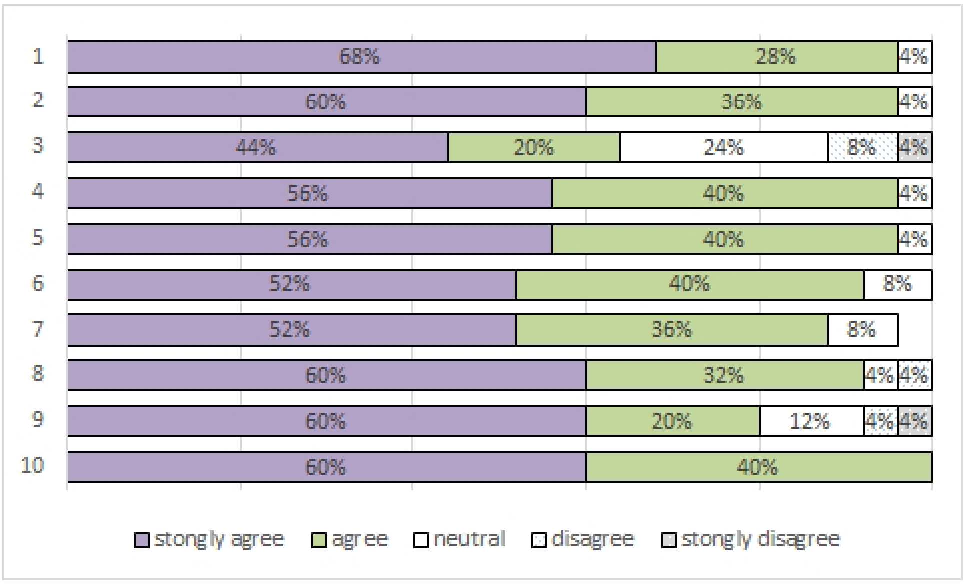


Figure 3: Direct participation (student pair) N=25

Passive method: viewing the video of tutors pair programming

Nine of ten items had a median value of ``agree''; item 3 was the exception: ``I feel connected to others in this course'', had a median value of ``neutral'' (N=34). An explanatory comment made by one participant to accompany item 3, connectedness, was ``I rarely have contact with anybody on the course''.

Indirect participation method: watching tutors pair programming live

As with the passive method, nine of ten items had a median value of ``agree'', whereas the median value of item 3 was mid-way between ``agree'' and ``neutral'' (N=22). This indicated a small increase in a feeling of connectedness with others in course: as one participant added to the survey response, ``It was good to see so new faces and hear their questions and comments.''

Direct participation: pair programming with an online partner

Nine of ten items had a median value of ``strongly agree''; for the tenth, item 3, there was a further increase in a feeling of connectedness, with a median value of ``agree'' (N=25). Four comments added to the survey responses confirmed positive feelings, such as ``Was a great idea. Allowed me to connect on a personal level by chatting socially as well as regarding the task at hand" and ``pair programming with an actual student made me feel very connected". Interestingly, the event appears to have prompted one student to connect further with peers: ``I have been using a social program called discord to speak with the other students **since taking part in the task**" (emphasis added). In contrast, one student concluded that pair programming was not a preferred mode of working: ``i think i prefer to work in isolation or **with people I am familiar with** (emphasis added).

Figures 1--3 illustrate that increases in the extent of directness in participation by the students are associated with increases in their feelings of agreement with all items in the survey. These findings were then explored via a small focus group.

2. Focus group
Five focus group topic areas were identified from the survey data:

- feelings (how did participating in the project make you feel?)
- learning (what did you learn about yourself?)
- looking ahead (how should this project be taken forward?)
- logistics (how did you make decisions with your partner?)
- conclusion (of all the things discussed, what to you is the most important? is there anything else you want to share?)

Comments were compared and coded, codes were reviewed and revised, and from these a set of themes were confirmed (Table 1).

Table 1. Focus group analysis

Comments	**Themes**
reassurance, review, feedback, mentoring, fine-tuning, share ideas for resources, discuss and get inspiration when stuck, extra exposure to the topic, each modality had its place	how it helped
positive, helpful, valuable, good, enjoyable, collaboration, communication, interaction, learned, listen to other ideas, helped others to learn, learn to ask for help, social discussion, social dimension, social confirmation, echoing	benefit
screen sharing, too simple scenarios, disparity of experience, put on the spot, performing less than storming/norming, sessions too few, too far apart	downside
more real-world scenarios, more sessions, shorter gap in between, hackathon or group session, swapping to different pairs	suggestion

Most comments from which the themes were drawn were self-explanatory. Deserving of additional explanation is the comment that each modality has its place. One student made this comment on more than one occasion, emphasising its importance to him:

> *"I think the pair programming where you're actually on a one to one with somebody else has its place but I think the other two modalities where you watch a video recording and you could have an option and listen to a bit again, the middle situation where you're attending a webinar and you can ask questions through the chat box. So, I think they all have their role."*
>
> *"I think the initial video, you know, is helpful to see how two professionals might do it and ... if it's already recorded you have the opportunity to go back and if there's something you're not quite sure of you can always replay the bit again. And then on live you can put in the web chat. But obviously actually doing it there's nothing like a live situation'"*

Comments about learning to ask for help were noted by one student as useful for the development of an employability skill:

> *"There is a lot of scope for a lot of thoughts filling your head and so getting used to and indeed experiencing your first kind of asking a colleague for that kind of help is quite useful in an employability sense and it makes you a better employee to be able to struggle and ask."*

One pair identified a difference in the level of their skills. The less experienced student described himself as the `"back-seat driver'" rather than the navigator. That student recognised benefit to him of the pairing but was concerned on behalf of his partner:

> *"when I was stuck, A was there to provide inspiration. I don't know how much A found that situation beneficial"*

Reassuringly, student A had identified different benefits for himself, and responded with observations about mentoring:

> *"to mentor is also an essential part of being a programmer cos that's when you get a lot of the rough corners and a lot of the funny questions asked and you learn a lot from mentoring, as much as you do from being mentored."*

Discussion

1. Survey findings

Passive participation

Results indicate that even the apparently passive process of watching a video of two tutors pair programming was perceived positively by the majority of students for all items, except for item 3: I feel connected to others in this course. However only one quarter of the students, approximately, felt that watching a video helped develop connectedness. For those students, there may have been some revelation of the tutors' personalities. Possibly seeing the tutors' hesitations and occasional doubts (or trivial programming errors) did help some students to recognise that everyone has the same human fallibility, and so the perception of connectedness was with the tutors as well as with their peers. A relevant comment made by one participant demonstrated some feeling of familiarity with the two tutors in the video: ``In the video Brendan and Bobby were clear with their commutation (sic), I feel it's good example and believe I could work with someone as easily as they have''. Watching a video of tutors pair programming could be described as an example of vicarious learning, which has been recognised for some decades since the concept was first introduced by Bandura (1965). Stenning et al. (1999) have proposed that learning can occur by observing others participate in dialogue.

It is also possible that some students appreciated actually hearing the tutors pronounce the code: Hermans et al. (2018) reported that novices may have additional cognitive load as a result of the fact that vocalisation of code is not standardised. One of their examples is the pronunciation of an assignment statement like x = 5: ``is it "x is 5"? Or "set x to 5"? Or "x gets 5"?''; they advocate the development of consensus about how code snippets should be pronounced, which they also speculate may help pair programmers to communicate (Hermans et al., 2018). In related work, Swidan and Hermans (2019) found that reading code aloud could contribute to improved code comprehension for primary school children learning to code.

Clearly very many higher education institutions have embraced the use of video for online teaching during the pandemic circumstances, amongst other digital resources. There is already evidence that the use of digital resources has had some positive impacts; as an example, Pochino et al. (2020) described a study of over 200 teachers in one European country which found a number of positive impacts of the use of digital resources in online teaching during the pandemic; student social experiences and community did not feature, however, and notably the authors describe the period as being one of social detachment, and social distance. Veldthuis et al. (2020) compared flipped classroom teaching in 2019 with that in 2020, when it had to become fully remote; they concluded that preparing high quality lecture materials can be successful, but in their study these are accompanied by other mechanisms for student discussion and engagement. In general, videos can be time-consuming to produce well, and their use without additional support for student-student interactions neglects the non-technical aspects of the students' higher education experience.

Indirect participation

Using Microsoft Teams to watch two tutors pair program remotely in real-time was perceived even more positively by the student participants. Notably stronger agreement was in the perceptions of decision-making ability (item 5) and confidence (item 7). One observation made in comment about the indirect participation method suggests that participants may have been checking and testing the approach or decisions they would have taken themselves, and comparing these to the tutors' actions, thereby increasing their confidence:

> "*I was programming the solution along with the presenting pair. So it was interesting to see where I started and where they started*".

This comment resonates with early work by Sutton (2001) who proposed that students who observe and actively process interactions between others can benefit by means of the vicarious interaction.

Using this indirect participation method, even the connectedness issue was perceived more positively. This is likely to be a result of the students seeing, hearing and 'chatting' to each other at the start and at the end of the pair programming session, and reading the comments and questions typed by individuals during the session. Students would have gained further familiarity with their tutors, further recognising their different personalities and programming approaches.

To increase students' confidence with interacting in such a circumstance, educators may need to provide advice and encouragement about interrupting and querying. Recent work by Mardi et al. (2021) suggested a novel way to guide new students towards successful dialogue during pair programming, which they described as pair coaching: an expert and a novice were observed pair programming in real-time by the students. Since the novice was also an educator, her questions were delivered confidently, at times helping the expert to realise that his explanations were less than perfect. The students were able observe the technique of interacting in a way that was, as Mardi et al. (2021) summarised, ``being intellectually disruptive in a constructive way''.

Direct participation

Most evidently appreciated by the participants was direct participation, that is the experience of actually pair programming with one of their peers. Predictably, the direct learning process was perceived by participants to be more useful than the vicarious and indirect methods. Bonwell and Eison (1991) described active learning as ``instructional activities involving students doing things and thinking about what they are doing''. Multiple authors since have provided evidence of the effectiveness of active learning, e.g. as summarised by Prince (2004) and in Gonzalez' (2006) study which found that more CS1 students with an active learning experience had passing grades and fewer withdrew than those with a regular learning experience. The survey responses in this study showed that virtually twice the percentage of participants were strongly in agreement with each one of the item statements for this method compared to the other two methods. Strongly agree and agree responses had increased predominantly at the expense of neutral responses. Even item 3, connectedness, was markedly different in this method compared to the other two methods: the majority of students agreed for the first time that they felt connectedness to others in the course.

Whilst there was greater agreement for all the items, those which were notably more strongly agreed with by means of the direct participation method were verbal communication (item 5), decision-making (item 4), and initiative (item 6). An interesting observation by one student after pair programming with a partner: ``Having done this exercise, I now think I have points to work on to improve communication''. Improved confidence with verbal communication itself may be contributing to improved confidence with decision-making and taking initiatives. Rodriguez et al. (2017) investigated different aspects of effective pair programming collaboration in a study of undergraduate computer science students, and concluded that the best outcomes for the students resulted from lively and substantive content-related talk between the pairs -- and noted that self-explanations and think aloud by drivers, plus active feedback from navigators, should be encouraged. Self-explanations have been recognised as beneficial by numerous researchers since the seminal work by Chi et al. (1994), who also credited the early findings by Webb (cited in Chi, 1994) that "giving elaborate explanations was positively related to individual achievement"; Chi et al. commented that "the advantages gained by explaining to others and to oneself are comparable". Zarb and Hughes (2015) found that communication guidelines can ease students towards successful interactions with their pair programming partner.

2. Focus group findings

The focus group codes and themes identified in this work correspond reasonably well to those described by Celepkolu and Boyer in their 2018 work analysing students' reflections on pair programming in CS1. Examples of such correspondences are:

- listen to other ideas as an example of ``exposure to different ideas''
- learn to ask for help as an example of ``deeper learning''
- social dimension as an example of ``social growth''
- disparity of experience as an example of ``partner''
- positive, helpful, valuable, good, enjoyable as examples of ``satisfaction''.

Celepkolu and Boyer (2018) extended their thematic analysis to derive three dimensions that encompassed all seven of their themes: cognitive, social and affective. Each of these dimensions has at least one correlating code in the responses given in this focus group.

Key issues identified via the focus group are summarised as follows:

1. all three methods of experiencing pair programming provided some benefit
2. positive experiences greatly outnumbered the drawbacks
3. positive experiences reported correspond to those reported in previous pair programming research: exposure to different ideas, deeper learning, social growth and satisfaction
4. downsides mostly related to event logistics
5. even with a poorly-matched pair, other benefits were identified (learning to ask for help, mentoring).

There are similarities here with the findings of VanDeGrift (2004) whose students did not all live on campus: they identified the social nature of the pair programming work as one of the benefits but noted that the logistics of scheduling time for meetings was problematic; unlike in that study, the students in this instance did not report partner personality differences or skills levels as problematic

These results also are comparable to the recent findings by Ying et al. (2019) in a qualitative study in which 94\% of over one hundred introductory computer science course students introduced to pair programming reported at least one positive sentiment about the experience. Their thematic analysis reported benefits such as improving learning experience (including broadens problem-solving techniques), career skills (including learn to work together), positive atmosphere, networking (including social/communication skills), more engaged and personal gain (including builds confidence) (Ying et al., 2019),

Limitations

A number of limitations existed in this project. In the first instance, the students were self-selecting: they may have been hoping for benefits rather than being completely objective. Secondly, their responses to the items reflect their perceptions, which are not necessarily correct. Since Likert-type items were used, it must be recognised that the participants may have preferred to respond in the central categories (although a small number of responses were “strongly disagree” and a large number were “strongly agree”, and so this limitation may not apply). However they may have preferred to agree with the statements given. Thirdly, the sample size was not large and the methods were not of an extended duration. Students could watch the video as many times as they wished, and were at liberty to arrange as many further pair programming meetings with their partner as they wished, privately, for a period of three more months - but the indirect participation method will have been only experienced once. Furthermore, the survey and focus group were not repeated with a similar group of students who were learning to program solo, from home, without any of the pair programming experiences at all; therefore there is no benchmark against which these results can be compared. Finally, this work was designed to understand the experiences of the participants as they felt it; it is unlikely that the situation and its interactions could be entirely replicated, and therefore the results cannot be generalized to a wider context with dissimilar samples.

Main findings

This project aimed to investigate if distance learners experience non-technical benefits from participating in remote pair programming. Student participants experienced three different methods of pair programming, ranging from passive to direct involvement. Results from the survey and from the focus group suggest associated non-technical benefits, irrespective of the specific method experienced. Whilst all three methods of experiencing pair programming were reported positively by the participants, the direct participation method of working with a remote partner led to the strongest agreement that these benefits accrued, even when working with a student who was previously completely unknown. Pairing of students was achieved on the basis of their previous experience, their availability, and any special issues disclosed. Only one pair of students reported any mismatch of ability level, but still these students found value in the interaction.

Overall, benefits perceived included improvements in verbal communication, problem solving, initiative and decision-making. Interactions were found to be good in terms of the social experience, for confirming technical understanding, for gaining inspiration, for learning how to ask for help and for developing the skill of mentoring. The social and community benefits of remote pair programming perceived by participants were similar to some reported previously as experienced when programming face-to-face with a partner; Ying et al. (2019) for example, included improved learning experience, networking and a positive atmosphere amongst positive perceptions of pair programming.

The findings of this project are relevant, given the current importance of optimising remote teaching and learning in the circumstances of locked-down education. Educators are aware that many aspects of student life contribute to the student experience in higher education: social and support aspects can be crucial to student retention, and academic progress may depend upon mental welfare as much as high quality teaching. Therefore designing into the curriculum opportunities for distance learning students to interact with tutors and with each other should contribute to the sense of community and opportunity for collaboration, as well as further develop some of the soft skills of the type valued by employers.

In the pre-pandemic software development world, programmers could be working simultaneously in the same room or building, conveniently sharing questions and suggestions. New employees could gain considerably from supportive colleagues nearby. There is considerable speculation that employment in the post-pandemic world will not have the same office-based style of working, particularly if employers believe that home working can minimise costs and optimise efficiency. That being the case, new graduates will have considerable need of remote working communication and collaboration skills as experienced here in remote pair programming if they are to avail themselves of colleague support networks. Direct participation remote pair programming can help students to develop the social confidence to discuss their programming with new acquaintances. The indirect participation method can help students to regard their more experienced tutors as approachable, learning how to break the social barrier and/or discomfort of engaging with more experienced or senior others to articulate their questions. In these ways at least, we believe that non-technical remote pair programming skills can benefit organisations as well as individuals.

In the context of this project, the responses from the student participants suggest a positive view of the original speculation: is the effort of arranging remote pair programming worthwhile? Remote pair programming can supplement the social and community aspects of the student experience that are otherwise limited for distance learners, even when students are partnering with someone whom they have never met, and it may help prepare students for the non-technical aspects of the post-pandemic world of home working.

Recommendations

Recommendations for module teams and tutors designing programming teaching for OU students are:

- Prioritise real-time interactions rather than devoting lots of attention to the creation of faultless demonstration videos: considerably more value will come for students' sense of community by means of them experiencing authentic real-time events.
- Focus on preparing the students for communication, turn-taking, interacting, and questioning rather than the process of using the technologies: they need support to develop their confidence establishing a social dynamic in a non-solitary environment.
- Aim to match student pairs according to expertise and self-efficacy, but also establish and take account of any expressed preferences for partnership (for example regarding the gender of the partner) or special needs, such as anxiety about skill or level of English-language.
- Encourage students to appreciate the benefits that can be gained when partnering up people even of different levels of expertise, particularly in the areas of desirable soft skills, such as articulating their own reasoning, learning to ask for help, and mentoring.

Future work

Our findings have been shared with colleagues leading relevant modules, in the hope that they may wish to extend the remote pair programming into other programming courses. It is believed that the benefits of remote pair working should be considered in other (non-programming) modules which may benefit from greater communication and collaboration, such as database management and analysis, or information security. A future project might be to develop guidance for students specifically to help the development of the professional and sensitive skills of querying, challenging and defending their own work. Further work is needed to examine the connectedness issue to understand what is needed to support it further.

Finally, we believe the OU should continue to explore alternative technologies for remote pair programming, such as Microsoft LiveShare (https://code.visualstudio.com/learn/collaboration/live-share) and Replit (https://replit.com/). Whilst many students can find ways to succeed with different technologies for pair programming, technological hurdles may be especially off-putting for distance learning students working solo from home, e.g. Adeliyi et al. (2021). A tricky interface or process may be sufficient barrier to others to continue, similar to the communication problem described by Canfora et al. (2003) as one of four causes of pair dismissal.

Impact

a) Student experience

- *In what ways has your project impacted on student learning?*
- *How is your project contributing to increasing student success (i.e. retention, employability, etc.)?*
- *Have there been or will there be any benefits to students not directly involved in your project?*

Our work on this project has increased our awareness of the benefits that students can gain from working together with their fellow students. We have discovered new possibilities for supporting our students and helping them gain employability skills such as working in a team and learning how to ask for help.

The YouTube video [(1) Pair programming research project: TM112 20D - YouTube] has had 102 views.

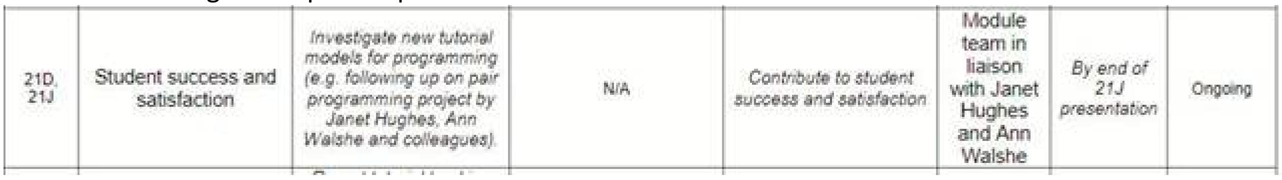
b) Teaching

- *How have you affected the practice of both yourself and others within the OU?*

Our project prompted discussions with academic colleagues including a postgraduate student who is investigating tools for pair programming, and the student's supervisors, leading to us sharing literature references, ideas, and jointly presenting a conference poster.

There is ongoing work within TM112, with further work planned in Scotland by the project team members. Other colleagues in the School of Computing and Communications are interested in investigating further the tools that enable pair programming, and are planning to run some pair programming sessions on other modules that include programming, such as M269 which uses programming as a tool for algorithmic problem solving.

The TM112 rolling action plan as presented to C&C Board of Studies in October 2021 includes:



21D, 21J	Student success and satisfaction	*Investigate new tutorial models for programming (e.g. following up on pair programming project by Janet Hughes, Ann Walshe and colleagues).*	N/A	*Contribute to student success and satisfaction*	Module team in liaison with Janet Hughes and Ann Walshe	*By end of 21J presentation*	Ongoing

In future, it may be worth considering a training event on how tutors can use shared programming screens especially in drop-in sessions (or virtual computing labs).

We believe that the project has contributed to professional development of all members of the project team with respect to the research topic, ethical procedures and in general it has enlarged the team's appreciation of the benefits to the OU of the eSTEeM initiatives.

- *What has been the impact of your project outside the OU?*

The Covid situation has made our project even more relevant to outside institutions as they moved to online teaching. We were able to discuss our project with participants at several conferences.

Our CSEDU paper has been cited by a set of researchers in Linkoping University, Sweden.

Our SIGCSE poster has been downloaded 91 times, including 10 times in the last six weeks (as of October 2021).

We shared the video (with Brendan and Bobby's permission) upon request with Glasgow Caledonian University, with feedback from the AL who also teaches there as follows:-

> *I do not have any specific feedback as such. I cover the topic of pair programming as part of my discussion about the use of agile methods on the 1st year module fundamentals of software*

engineering. I try very hard to relate what I am delivering to the programming that the students do on the module programming one but there is very little opportunity to investigate pair programming as such.

I put the link to the video up as part of the preparation and as feedback on what pair programming was about but I have not had any specific comments from students. I think there is scope for using the video more specifically in relation to what pair programming is and for following this up with a couple of tutorial questions. There may be scope for doing this in the next presentation of the Fundamentals of Software Engineering starting in May.

I would be interested in hearing how you are using/planning to use the video on TM112. My one observation is that it is a bit long at 29 minutes. I think that in principle this is a good resource that illustrates the idea of collaboration very well.

c) Strategic change and learning design

- *What impact has your work had on your Unit's or the University's policies and practices?*

d) Any other impact

- *For example, secured additional external funding.*

List of deliverables

Workshop: : Janet Hughes, Bobby Law, Brendan Murphy, Ann Walshe (2021) **Remote pair programming** at 10th eSTEeM annual conference 30th June – 1st July 2021, Online

Adeliyi, Adeola; Hughes, Janet; Kear, Karen; Law, Bobby; Murphy, Brendan; Rosewell, Jon; Walshe, Ann and Wermelinger, Michel (2021). **Remote Pair Programming.** In: *SIGCSE'21 - Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, 13-20 Mar 2021, Virtual, ACM.

Hughes, Janet; Walshe, Ann; Law, Bobby and Murphy, Brendan (2020). **Remote Pair Programming.** In: *13th International Conference on Computer Supported Education*, 02-04 May 2020, Online (originally Prague, Czech Republic), pp. 476–483.

Poster presentation: Janet Hughes, Bobby Law, Brendan Murphy, Ann Walshe (2019**) Online pair programming: Benefits to distance learning students** at ninth Pan-Commonwealth Forum on Open Learning (PCF9), 9-12 September 2019, Edinburgh https://www.col.org/about/pan-commonwealth-forum/ninth-pan-commonwealth-forum-open-learning-pcf9

Figures and tables

References

Adeloa Adeliyi, Janet Hughes, Karen Kear, Bobby Law, Brendan Murphy, Jon Rosewell, Ann Walshe and Michel Wermelinger (2021). Remote Pair Programming. In: SIGCSE'21 - Proceedings of the 52nd ACM Technical Symposium on Computer Science Education, 13-20 Mar 2021, Virtual, ACM.

Vasa Buraphadeja and Jirang Kumnuanta. 2011. Enhancing the sense of community and learning experience using self-paced instruction and peer tutoring in a computer-laboratory course. Australasian Journal of Educational Technology 27, 8 (2011).

Gerardo Canfora, Aniello Cimitile, Aaron Corrado Visaggio. 2003. Lessons learned about distributed pair programming: what are the knowledge needs to address? In: WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 314-319.

Mehmet Celepkolu and Kristy Elizabeth Boyer. 2018. Thematic analysis of students' reflections on pair programming in cs1. In Proceedings of the 49th ACM Technical Symposium on Computer Science Education. 771–776.

Michelene TH Chi, Nicholas De Leeuw, Mei-Hung Chiu, and Christian LaVancher. 1994. Eliciting self-explanations improves understanding. Cognitive science 18, 3 (1994), 439–477.

Soroush Ghorashi and Carlos Jensen. 2016. Jimbo: a collaborative IDE with live preview. In Proceedings of the 9th International Workshop on Cooperative and Human Aspects of Software Engineering. 104–107.

Brian Hanks, Charlie McDowell, David Draper, and Milovan Krnjajic. 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). ACM, New York, NY, USA, 176–180.

Jo E Hannay, Tore Dybå, Erik Arisholm, and Dag I. K. Sjøberg. 2009. The Effectiveness of Pair Programming: A Meta-Analysis. Inf. Softw. Technol. 51, 7 (July 2009), 1110–1122.

Janet Hughes, Ann Walshe, Bobby Law, and Brendan Murphy. 2020. Remote Pair Programming. In 12th International Conference on Computer Supported Education, CSEDU 2020.

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

Sandeep Kaur Kuttal, Jarow Myers, Sam Gurka, David Magar, David Piorkowski, and Rachel Bellamy. 2020. Towards Designing Conversational Agents for Pair Programming: Accounting for Creativity Strategies and Conversational Styles. In 2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). IEEE, 1–11.

Teerapong Leelanupab and Tiwipab Meephruek. 2019. CodeBuddy (Collaborative Software Development Environment): In- and Out-Class Practice for Remote Pair-Programming with Monitoring Coding Students' Progress. In Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19). Association for Computing Machinery, New York, NY, USA, 1290.

Colleen M. Lewis and Niral Shah. 2015. How Equity and Inequity Can Emerge in Pair Programming. In Proceedings of the Eleventh Annual International Conference on International Computing Education Research. Association for Computing Machinery, New York, NY, USA, 41–50.

Fatemeh Mardi, Keith Miller, and Phyllis Balcerzak. 2021. Novice - Expert Pair Coaching: Teaching Python in a Pandemic. In Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE '21). Association for Computing Machinery, New York, NY, USA, 226–231.

Jonathan McKinsey, Samuel Joseph, Armando Fox, and Daniel D Garcia. 2014. Remote pair programming (RPP) in massively open online courses (MOOCs). In Proceedings of the 2014 conference on Innovation & technology in computer science education. 340–340.

Ricardo Pocinho, Pedro Carrana, Cristov aoMargarido, Rui Santos, Sandrina Milhano, Bruno Trindade, and Gisela Santos. 2020. The Use of Digital Educational Resources in the Process of Teaching and Learning in Pandemic by COVID-19. In Eighth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'20). Association for Computing Machinery, New York, NY, USA, 810–816.

Fernando J Rodríguez, Kimberly Michelle Price, and Kristy Elizabeth Boyer. 2017. Exploring the pair programming process: Characteristics of effective collaboration. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education. 507–512.

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

Till Schümmer and Stephan G Lukosch. 2009. Understanding tools and practices for distributed pair programming. Journal of Universal Computer Science, 15 (16), 2009 (2009).

Alan Clinton Shaw. 2009. Extending the pair programming pedagogy to support remote collaborations in cs education. In 2009 Sixth International Conference on Information Technology: New Generations. IEEE, 1095–1099.

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

Karthikeyan Umapathy and Albert D. Ritzhaupt. 2017. A Meta-Analysis of Pair-Programming in Computer Programming Courses: Implications for Educational Practice. ACM Trans. Comput. Educ. 17, 4, Article 16 (Aug. 2017).

Tomoyuki Urai, Takeshi Umezawa, and Noritaka Osawa. 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. 177–182.

Marcella Veldthuis, Hani Alers, Aleksandra Malinowska, and Xiao Peng. 2020. Flipped Classrooms for Remote Teaching during the COVID-19 Pandemic (CSERC '20). Association for Computing Machinery, New York, NY, USA, Article 16.

Linda L. Werner, Brian Hanks, and Charlie McDowell. 2004. Pair-Programming Helps Female Computer Science Students. J. Educ. Resour. Comput. 4, 1 (March 2004), 4–es.

Laurie Williams and Richard L. Upchurch. 2001. In Support of Student Pair-Programming. SIGCSE Bull. 33, 1 (Feb. 2001), 327–331.

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

Krissi Wood, Dale Parsons, Joy Gasson, and Patricia Haden. 2013. It's never too early: pair programming in CS1. In Proceedings of the Fifteenth Australasian Computing Education Conference-Volume 136. 13–21.

Kimberly Michelle Ying, Lydia G. Pezzullo, Mohona Ahmed, Kassandra Crompton, Jeremiah Blanchard, and Kristy Elizabeth Boyer. 2019. In Their Own Words: Gender Differences in Student Perceptions of Pair Programming. In Proceedings of the 50th ACM Technical Symposium on Computer Science Education. Association for Computing Machinery, New York, NY, USA, 1053–1059.

N. Z. Zacharis. 2011. Measuring the Effects of Virtual Pair Programming in an Introductory Programming Java Course. IEEE Transactions on Education 54, 1 (2011), 168–170.

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

University approval processes

If your project required specific approval from university committees, please provide the appropriate information below. This is a necessary requirement for future publication of outputs from your project.

- *SRPP/SSPP – Approval from the Student Research Project Panel/Staff Survey Project Panel was obtained according to the Open University's code of practice and procedures before embarking on this project. Application number SRPP2019-125*
- *Ethical review – An HREC-Project-Registration-and-Risk-Checklist was included in the documentation submitted to SRPP. The project was approved with no further ethical review required.*
- *Data Protection Impact Assessment/Compliance Check – The Data Protection Impact Assessment (DPIA) Screening questions were completed and determined that a DPIA was not necessary.*