

# An Investigation of Gender Bias in Pair Programming

*Asli Akalin*



Electrical Engineering and Computer Sciences  
University of California, Berkeley

Technical Report No. UCB/EECS-2023-155

<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-155.html>

May 12, 2023

Copyright © 2023, by the author(s).  
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

### Acknowledgement

I would like to express my most sincere gratitude to my advisor and mentor Armando Fox for giving me the opportunity to join ACE lab and Fox group as without his guidance and support in the last few years I wouldn't be here.

I also would like to thank Nate Weinman for being an amazing mentor as well as Katherine Stasaski, who both gave valuable feedback during the entire process of developing the user study, and devoted many hours running the pilot sessions with me.

I also would like to recognize the work of University of Seville team: Pablo, Amador, Bea, and David. Their work have been instrumental in developing Twincode and Tag-a-chat platforms, running the primary study in Seville and coauthoring the EMSE paper which reports on the results of both Seville and Berkeley studies.

---

# An Investigation of Gender Bias in Pair Programming

by Ashi Akalm

---

## Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences,  
University of California at Berkeley, in partial satisfaction of the requirements for  
the degree of **Master of Science, Plan II.**

Approval for the Report and Comprehensive Examination:

**Committee:**



---


Professor Armando Fox  
Research Advisor

**May 5, 2023**

---

Date

\* \* \* \* \*



---

Professor Joshua Hug  
Second Reader

**12-May-2023**

---

Date

# Contents

<b>1</b>	<b>Overview and Motivation</b>	<b>5</b>
<b>2</b>	<b>Research Questions and Study Design</b>	<b>6</b>
2.1	Research Questions . . . . .	7
2.2	Twincode and Tag-a-Chat Platforms . . . . .	8
<b>3</b>	<b>Related Work</b>	<b>11</b>
<b>4</b>	<b>Experiments</b>	<b>14</b>
4.1	Pilot Studies . . . . .	14
4.2	Recruiting and Participants . . . . .	18
4.3	Study Workflow . . . . .	24
<b>5</b>	<b>Analysis: Data and Variables</b>	<b>31</b>
5.1	Perception: Self-Reported Surveys . . . . .	32
5.2	Behavior: Twincode Platform Logs . . . . .	33
5.3	Communication: Chat Logs . . . . .	34
<b>6</b>	<b>Results</b>	<b>35</b>
6.1	Correlation of Induced and Perceived Gender . . . . .	35
6.2	Between-groups Analysis . . . . .	36
6.3	Within-groups Analysis . . . . .	38
6.4	Discussion, Limitations, Threats to Validity . . . . .	40
<b>7</b>	<b>Future Work</b>	<b>42</b>

## Acknowledgments

I would like to thank Armando Fox for giving me the opportunity to join ACE lab and Fox-group and work on this project and express my most sincere gratitude for being my advisor and mentor, as without his guidance and support in the last few years I wouldn't be here.

I also would like to express my gratitude to Nate Weinman who has been an amazing mentor as well as Katherine Stasaski, both of whom not only created the idea of the gender deception experiment and allowed me to pick it up as my research project but also have given valuable feedback during the entire process of developing the user study, and devoted many hours running the pilot sessions with me.

I also would like to recognize the work of University of Seville team: Pablo, Amador, Bea, and David. Their work have been instrumental in developing Twincode and Tag-a-chat platforms, running the primary study in Seville and coauthoring the EMSE paper [18] which reports on the results of both Seville and Berkeley studies.

Also, big thanks to Karim and Daewon for their help in hosting and running big pair programming sessions as well as to all the volunteers who participated in the study. Last but not least, I would like to thank Soda Hall staff for letting me put flyers everywhere and temporarily littering almost every flat surface in an effort to recruit students for this study.

## Abstract

Women have historically been underrepresented in software engineering, due in part to an unwelcoming climate pervaded by the widely-held gender bias that men outperform women at programming. Pair programming is both widely used in industry and has been shown to increase student interest in computer science, particularly among women; however if the gender biases are also present in pair programming, its potential for attracting women to the field could be thwarted. We aim to explore the effects of gender bias in pair programming, specifically, in a distributed remote setting in which students cannot directly observe the gender of their peers. Using deception we study whether the perception of the partner, impacts the behavior during programming, the style of communication or the perceived productivity and technical competency of the partner depending on the perceived gender of their partner. To our knowledge, this is the first study specifically focusing on the impact of gender stereotypes and bias *within* pairs in pair programming.

We observed statistically significant effects with moderate to large sizes in four of the 45 dependent variables within the experimental group, comparing how subjects acted when their partners were represented as a man or a woman. When subjects perceived their partners as women, they deleted more characters in the source code window and they displayed lower frequency of informal utterances, reflections and yes/no questions while communicating, compared to when they perceived their partners as men. When partners perceived their subjects as men, they delete fewer source code characters and communicate using more informal utterances, reflections and yes/no questions. These results must be considered carefully because of the small number of subjects; more replications are needed in order to confirm or refute the results in the same and other computer science student populations.

# 1 Overview and Motivation

Pair programming is a process in which two individuals share a single computer to work on a single program. The pair can traditionally switch between roles as the driver and the navigator in certain intervals while collaborating on the same analysis, design, implementation, and test. While it has been used in the industry since 1970, with rise to popularity of the agile development and extreme programming [2], pair programming is increasingly prevalent in computer science education [29].

As a result there is a large literature [24] created over the last two decades studying pair programming in higher education, particularly in introductory programming courses. While studying pair programming, there are various metrics and factors that authors used to categorize different cooperation aspects and learning outcomes in different studies. These can be split into 2 main categories:

1. **Metrics measuring the effectiveness:** 4 main measures of effective pair programming outcomes are technical productivity (total time spent, skill/knowledge transfer, code accuracy/performance, amount of the task completed, number of solutions found) , program design (quality score, code coverage, expert opinion, number of passing tests, LOC, number of code defects), academic performance (class/exam/lab/quiz/assignment scores, retention rate, course completion rate), and satisfaction (enjoyment, social interaction, positive attitude towards cooperation, opting in next time)
2. **Factors impacting effectiveness:** Compatibility factors such as personality types (MBTI etc), documented or perceived skill levels, communication skills, confidence in one's code, self esteem in one's ability, gender, ethnicity, learning style, work ethic, time management ability, feel-good factor, type of role taken (driver, navigator) and type of task at hand (debugging, developing, refactoring)

Learning new concepts and solving programming questions are difficult for many introductory computer science students. Drop-out, failure and withdrawal rates in introductory courses of 33% or greater are not uncommon [4]. Throughout the literature, educational benefits of pair programming include increased success rates in introductory courses, success in later courses without difficulty switching to solo programming [35, 37, 27] increased retention in a computing related major [7, 34, 35], higher quality software (less complex and more cohesive solutions with fewer compiling errors and fewer bugs) [25, 34, 39, 5, 3], higher student motivation and confidence (in their solutions reliability and in finding bugs faster in the code) [34, 25, 16, 6, 45, 36, 48] and improvement in perceived learning outcomes including enjoyment [9, 48, 45, 34, 43, 23, 22, 37]). Therefore pair programming may be an effective way to address difficulties in introductory CS courses generally. Furthermore, female students in programming courses are frequently less confident than men, even when their actual level of competence is the same [32]. This leads them to conclude that they do not “belong” in computing, and they leave at higher rates than men. Because fewer women than men attempt computing-related majors in the first place, this higher drop rate results in a greater gender gap in computing degrees. Luckily, though, the studies also suggest that pair programming can help increase retention rates and positively influence class performance, confidence and student motivation especially for women [40, 46]. This suggests that pair programming may have a positive impact to reduce the

gender gap in computing, if we can maximize the benefits and remove any potential hindrances for women in pair programming practices. The pair’s ability to engage and collaborate with each other without being influenced by bias is key to unlocking the benefits for all participants.

In this study we aim to explore the effects of gender bias [26, 33] in pair programming, analyze the impact of perceived gender (separated from actual gender) on both the productivity of and interactions between participants. Specifically, in a remote setting in which students cannot directly observe the gender of their peers. We study whether the perceived productivity, perceived technical competency of the partner, and collaboration behaviors differ depending on the perceived gender of their remote partner, (i.e scoring men and women differently on similar tasks, communicating using different patterns or changing their collaboration behaviors between a man and woman partner) even though their partners remain the same across pair programming sessions.

## 2 Research Questions and Study Design

Inspired by Whiting et al[47] experiment, we have undergraduate Computer Science students solve programming questions in a distributed pair programming setting. They only communicate through text-based chat and a shared editor. Although participants believe they are paired with two different partners, in reality they will be paired with the same partner twice. In one of these sessions, one participant perceives their partner as the opposite gender. Perceived gender is communicated through gendered pseudonyms and avatars only to one partner (treatment group) while the other partner sees no gender identifications. The students are unknowingly paired with the same partner twice to solve coding questions in a fixed time. All experiments are conducted under an approved IRB protocol and participants are informed of the true purpose of the experiment after completion.

The experiment consists of three programming sessions as shown in Figure 1: two pair programming sessions separated by one solo programming session. Although participants believe they are paired with different students each paired session, the same two students are paired together twice. In either session 1 or session 3 (randomly-selected), one participant will perceive their partner as the opposite gender. In order to remove other factors/confounds that may affect the pair programming experience, our study uses deception in a within-subject analysis to gain further insight into the impact of gender based bias. We limit all communications to a text chat interface with an infrastructure to have gendered names, avatars and pronouns displayed. To further ensure successful deception, we instruct participants from revealing identifiable information, and we hold an individual session between pair programming sessions to prevent participants from suspecting that their partner hasn’t changed. This methodology will help us determine the dimensions of gender bias, and specific behaviors that arise when the perception of gender changes. In a remote pair programming setting where partners can’t directly observe the gender of their collaborator, the goal of this study is to identify the effects of gender bias by observing the control group when their partner’s perceived gender changes. We use several methods to collect data from multiple sources and apply triangulation [17] like human’s intrinsically do [14]: (1) self-reported questionnaires to measure subject’s beliefs and perceptions, (2) logs from the Twincode app during collaborative code writing to measure



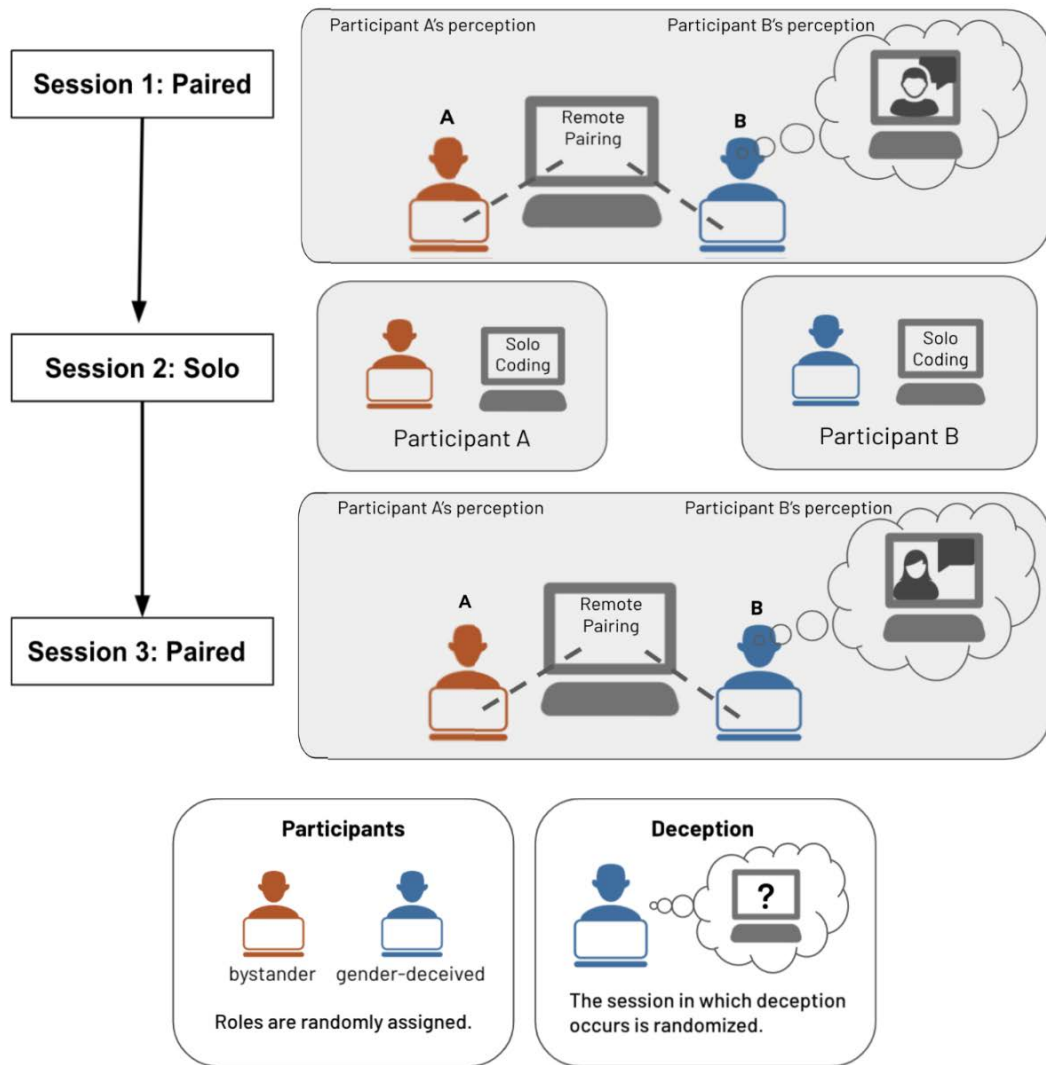


Figure 1: Diagram of the study consisting of three programming sessions.

behavioral changes, and (3) contents of the chat labeled with communicative labels as shown in Figure 5.

## 2.1 Research Questions

In this study we aim to answer the following questions with respect to the subjects' perceptions, behaviors and communications:

1. Does gender bias affect perceived productivity compared to solo programming? Do perceived differences between in-pair and solo productivity depend on the perceived gender of the partner?
2. Does gender bias affect the partner's perceived technical competency? Do perceived difference between the technical competency of one's own and partner's depend on the perceived gender of the partner?

3. Does gender bias affect the partner’s perceived positive and negative aspects? Do perceived positive and negative assessments depend on the perceived gender of the partner?
4. Does gender bias affect how partner’s skills are compared? Do perceived skills of the partner depend on their perceived gender?
5. Does gender bias affect the frequency of code additions, deletions, successful validations and chat utterance behaviors?
6. Does gender bias affect (relative) frequency of formal and informal chat utterances? Does formality of the communication depend on the perceived partner’s gender?
7. Does gender bias affect the frequency of the different types of chat utterances? Do the frequencies of the different types of messages depend on the perceived partner’s gender?

While we recognize that students may not identify as either men or women, our initial exploration focuses primarily on interactions between students who identify as men or women. The potential biases in interactions involving gender-fluid, gender-nonconforming and non-binary students is a complex topic deserving its own subsequent study.

## 2.2 Twincode and Tag-a-Chat Platforms

In collaboration with University of Seville we developed an online pair-programming platform called Twincode which allows students to participate in distributed pair programming sessions and work collaboratively through a shared code editor and limit communications to a text-based chat window. Twincode manages the subjects’ random (and balanced) allocation into treatment and control groups, the random allocation of pairs to match one control and one treatment group subject, the random ordering of programming exercises for each pair, the random swapping of the gendered avatars for the treatment group between pair programming exercises, and the collection of interaction metrics as logs as well as all messages in the chat window during collaboration. Figure 2 shows the platform during a paired session from the perspectives of control and treatment group subjects. Prior to the study Twincode presents a 5 minute practice exercise aimed at introducing the platform features and code editor step by step using the “Start Guide” button. During this practice exercise, the solution is given under the problem description so students can copy the correct solution and inspect test cases failing to pass while following the guide as shown in Figure 8. Partners can concurrently edit code and send messages, but to foster communication, only one partner can run the autograder tests at the same time. We instruct participants to iterate between navigator and observer roles and communicate test results to their partner using the chat window. Also note that the gendered avatar (as shown in Figure 4) and gender pronouns are only visible to one student per pair, whichever student has been assigned to the treatment group. Tag-a-Chat is the companion tool for twincode where the raw messages collected during the paired exercises can individually get labeled using the set of utterance tags as shown in Figure 3. It allows multiple labelers to work together and automatically computes agreement metrics such as Fleiss’s kappa (in our case

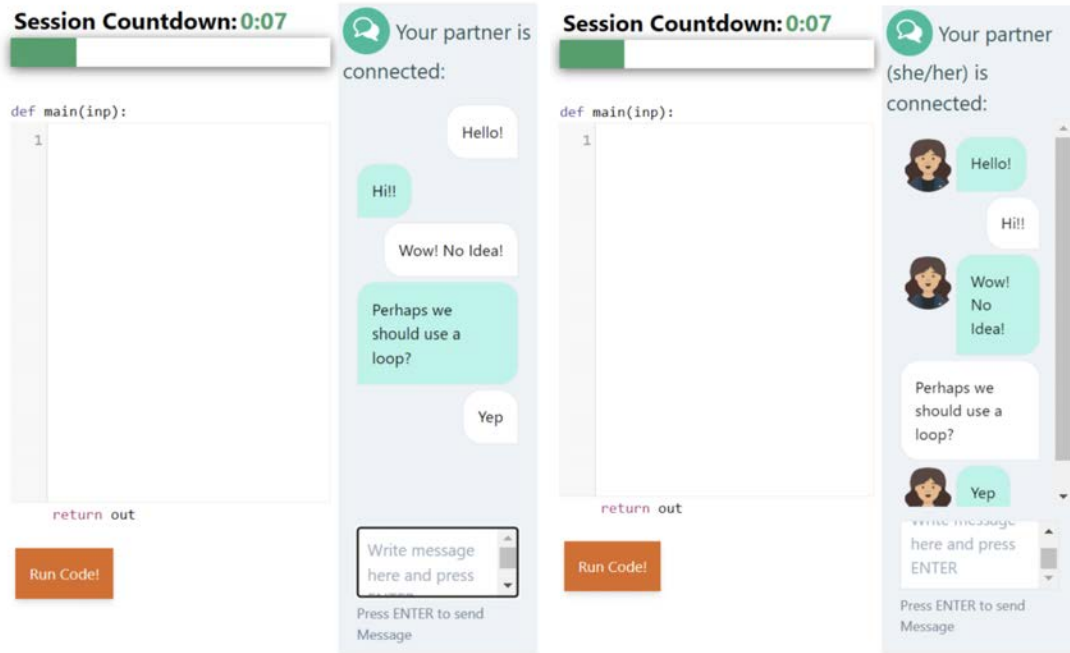


Figure 2: Twincode from the perspectives of control group (left) and treatment group (right). White messages are one's own, blue messages are of one's partner during the paired exercise. The chat window is disabled during the solo programming session.

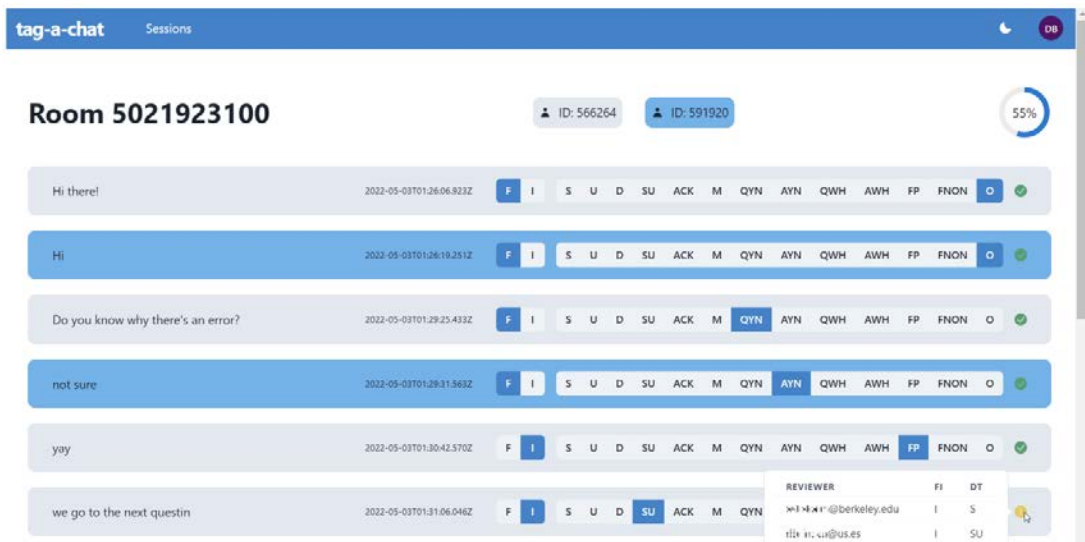


Figure 3: Tag-a-chat platform labeling messages into utterance categories.



Figure 4: Gendered avatars used in the chat windows of the subjects in the treatment group generated at <http://getavataaars.com/>. The treatment subjects were also shown “Your partner (he/him) is connected” or “Your partner (she/her) is connected.” above the chat window.

labeling was done by 3 organizers) to achieve inter-coder reliability assessment [38, 42].

To the best of our knowledge this is first study to measure bias within pairs. A run of the study using Javascript and in-person distributed pair programming setting was done in University of Seville. The study run in UC Berkeley differs in some aspects from that study: we ran the study in a remote setting during the COVID-19 lockdown over Zoom, rather than in a physical classroom. We recruited students towards the end of the semester using Python which required some tweaking of the platform and the questions from the Seville version.

### 3 Related Work

Considering the positive impact pair programming seems to have over learning outcomes, it is no surprise that it has been the subject of various studies over the decades, including its differential benefit for women.

Taking into account the high drop-out rates of computer science, frustration can be a key aspect when it comes to predicting retention rates. When frustration levels of students were investigated using a 5-point scale, the findings showed that solo females are more frustrated than solo males. When students were paired, female students who worked in pairs were less frustrated than working alone, but also, to emphasize the added benefit of pair programming for women, there was no difference in frustration between paired females and paired males [6]. These findings indicate that pair programming can benefit women more and be a great equalizer for frustration levels across male and female students.

In addition to lowering frustration levels, women’s impressions of the written reports (as a means of increasing confidence, technical understanding, and reflection) were higher than men’s for pair programming [45]. Female students indeed had more positive impressions of pair programming than men in multiple studies [45, 23, 46].

Paired women were more confident than solo women, and paired men were more confident than solo men. Moreover, pair programming had a greater impact on the confidence of women than that of men: Unpaired men indicated a confidence level 11.6 points higher than unpaired women, but for paired students the difference in confidence levels between women and men was only 3.5 points [46].

Women in pair programming were more likely than those who worked alone to take the final exam: 88.1% vs. 79.5%. There was also greater likelihood for women to have actually declared such a major 1 year after taking an introductory course if they were paired: for paired women, 55.5% were computing majors 1 year later, compared with only 22.2% of the women who had worked alone [46].

These findings suggest that pair programming may help reduce the amount of frustration experienced by female programmers and help increase female retention rates in computer science.

Interestingly, having more women in pair programming seems to benefit everyone. In a study using weekly attendance, and confidence metrics, it was found that students who were randomly assigned a woman partner attended class more often, and reported more confidence in the correctness of the finished product [28].

However, many factors may affect the outcomes of remote pair programming sessions [15].

One potential factor is implicit gender bias [26, 28, 33], which we will be focusing on in this report, such as assuming a female partner is less technically competent than a male partner [33]. This bias is a widely observed phenomenon even in highly-structured settings such as distributed pair programming [28, 15]. The presence of implicit gender bias in pair programming settings is worth investigating because social science research indicates that one’s behavior of an individual is affected by the behavior of their peers [19]. Therefore implicit gender-bias may have effects on one’s behavior based on how they are being perceived by peers, potentially influencing pair programming experience negatively for some women. Identifying and working towards removing such implicit gender bias can potentially attract women to the field and mitigate the gender gap in the field by tuning the pair programming process to maximize benefits for both partners.

Another challenge of “tuning” the pair programming process is pair compatibility. Previous research exploring what makes a pairing efficient and the session productive looked into factors such as skill levels [43], autonomy with choosing one’s partner [27] and different personality combinations [13, 7, 8] and gender [11, 10, 31, 26]. It seems that students are most compatible with perceived-to-be-similar skilled partners [30]. Following these results, we recruited similar-skilled students from introductory programming courses in CS61A and CS88 who have similar experience levels. The recruitment process is further detailed in “Recruiting and Participants” section.

Though various studies investigated various gender pairings the findings are hardly consistent as to whether mixed-gender or same-gender pairings are more compatible [12, 10, 26, 31, 30], possibly because gender correlates with other dimensions that may affect the pairs’ collaboration. Nevertheless, these studies were useful in highlighting some of the differences between men and women experiences in pair programming. For example, men were less aware and had milder beliefs of the gender disparities in computer science. Women were significantly more aware of the gender gap and felt significant efforts should be made to reduce the gender gap [50, 49]. Amongst productivity, quality of source code, compatibility and communication metrics, compatibility and communication levels significantly vary between same gender pairs (man-man and woman-woman) [12]. Across 3 gender compositions, productivity measurements were similar with a greater variability of productivity for mixed gender pairs [21]. Unlike previous findings which found no difference in productivity, one study also reported that being assigned a woman partner was associated with completing a smaller percentage of the assignment [28]. We combine qualitative self-reported survey results and quantitative Twincode metrics while evaluating productivity in our study.

Furthermore women reported higher levels of stress, lower levels of perceived confidence in their skills, and less perceived choice compared to the men as confirmed by their dialogue features [50]. Women seemed to be more relaxed if their partner used a positive tone of language and sent longer messages on average while communicating. To focus on these communication differences, one of the data sources we analyze in this study will be chat messages using the tags in Figure 5 below.

Tag	Description	Examples
I	Informal	<i>LOL! Hahaha!</i>
F	Formal	All messages except informal
S	Statement of information or explanation	<i>We need to create a program for kids to learn math</i>
U	Opinion or indication of uncertainty	<i>Unsure how to add strings together</i>
D	Explicit instruction	<i>Wait put the if back</i>
SU	Polite or indirect instruction	<i>Maybe we can do if user choice = +</i>
ACK	Acknowledgement	<i>Oh ok gotcha</i>
M	Meta-comment or reflection	<i>Hmmm</i>
QYN	Yes/no question	<i>Can the answer be negative?</i>
QWH	Wh- question (who, what, where, when, why, and how)	<i>How do I take in their input?</i>
AYN	Answer to yes/no question	<i>Yea</i>
AWH	Answer to wh-question	<i>The program should be able to generate erroneous questions</i>
FP	Positive task feedback	<i>Oh nice</i>
FNON	Non-positive task feedback	<i>Thats weird</i>
O	Off-task	<i>Wow its sweet in this room</i>

Figure 5: The methodology used to label chat messages as defined by [Rodriguez et al (2017)] consists of two parts: (i) first each message is categorized as either “formal” or “informal” then (ii) labeled with one of 13 tag categories indicating the content of the message such as: statement, uncertainty/opinion, acknowledgement, yes/no question, wh- question, meta comment, explicit instruction, polite instruction, positive or non-positive feedback, off-task comment, answers to yes/no and wh- questions.

## 4 Experiments

### 4.1 Pilot Studies

In order to get early feedback on the Twincode platform and test the full run of the study including participant instructions, questionnaires and programming questions, we ran pilot studies in December 2021 with 4 pairs [20]. Following these studies various changes were made on the platform, participant instructions as well as recruiting and execution of programming sessions. The final study with more participants was run in May 2022. This section reflects on the initial study execution, the feedback received and the improvements made leading up to the final study.

To monitor the interactions closely, the initial approach was to ask participants to share their screens over zoom call and record the sessions using the screen sharing feature. This meant that only one pair could go through the study at a time and it required 2 organizers to meet each participant individually on zoom to onboard them onto the platform. The survey links in between each programming session were sent over the zoom by the organizers. Similarly the initial and the final consent forms were shared with the participants as separate links when participants stated they were done.

As supported by literature, scheduling and attending the programming sessions were the biggest obstacle in running pilot studies. No contingency plan existed for last minute no shows: on three occasions the organizers had to act as the second programmer to handle such cases. Unfortunately, due to the automatic randomization of the platform we had no way of controlling whether we get assigned to the treatment or control group, thus rendering the pairs in which we were assigned to the treatment group unusable.

As reported by the students retroactively, some reasons for not attending the session were “worried about programming level” and “lack of confidence to solve interview style questions correctly” as well as “busy/changed [their] mind about the study”. There was one subject who left in the middle of the study without signing the consent form saying they disliked having their sessions recorded and it made them behave less authentically. Another participant went idle in the last 10 minutes (during the second paired exercises) because, the study went over the next hour mark due to their partner joining late. As a result none of the pilot pairs’ data were usable, but it helped us update the execution flow and presentation of the study.

To handle no-shows, we decided to host larger sessions with about 20 students rather than recording individual pair sessions. We increased the frequency of reminder emails coming up to the scheduled session, sent login codes to the platform and participant instructions prior to the study to give a sense of responsibility to the participants and also improved the reliability of the Twincode platform to collect more metrics and handle more participants rather than individual relying on the recordings for evaluation. We also offered \$15 Amazon gift cards to all participants who successfully complete the study as an incentive for participants to actually show up and complete the study which were sent after all participants participated in the study. Furthermore, in recruitment announcements and flyers we added an emphasis of “pair interaction and collaborative work” over “correctly solving the questions.” An example of such “friendlier” announcements is shown in Figure 7. These updates combined with a more aggressive recruiting campaign increased participation, attendance and follow-through significantly.



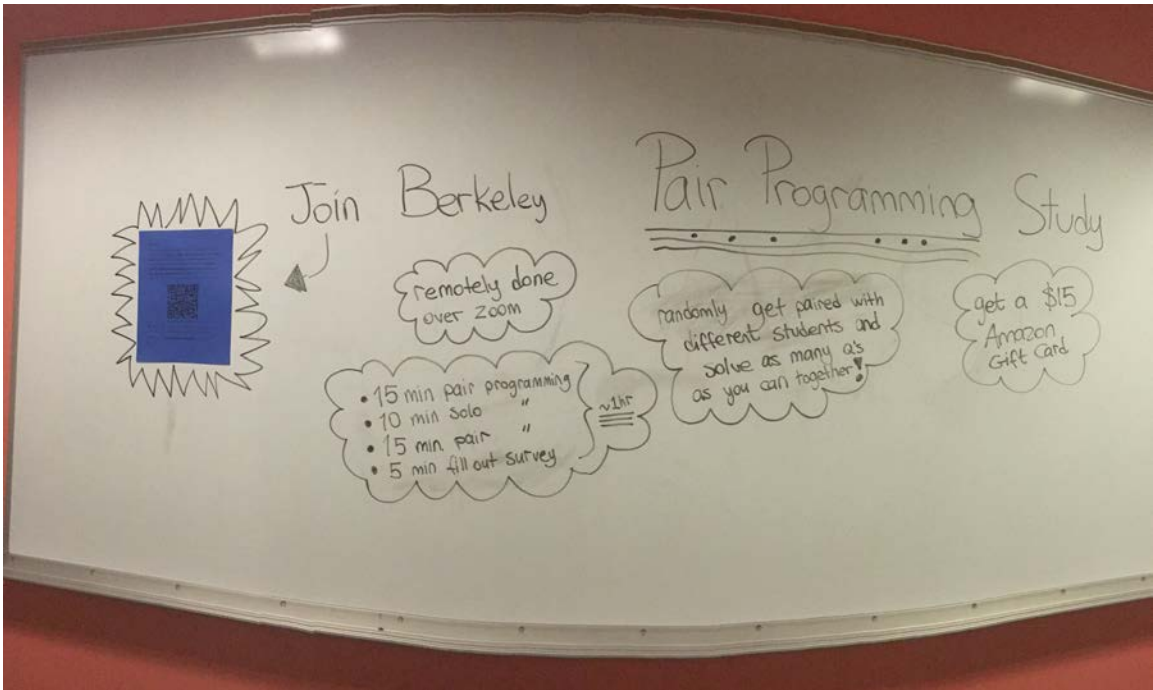


Figure 6: An announcement introducing the study written on public whiteboards in front of Soda Hall labs where introductory programming courses meet for lab sections.

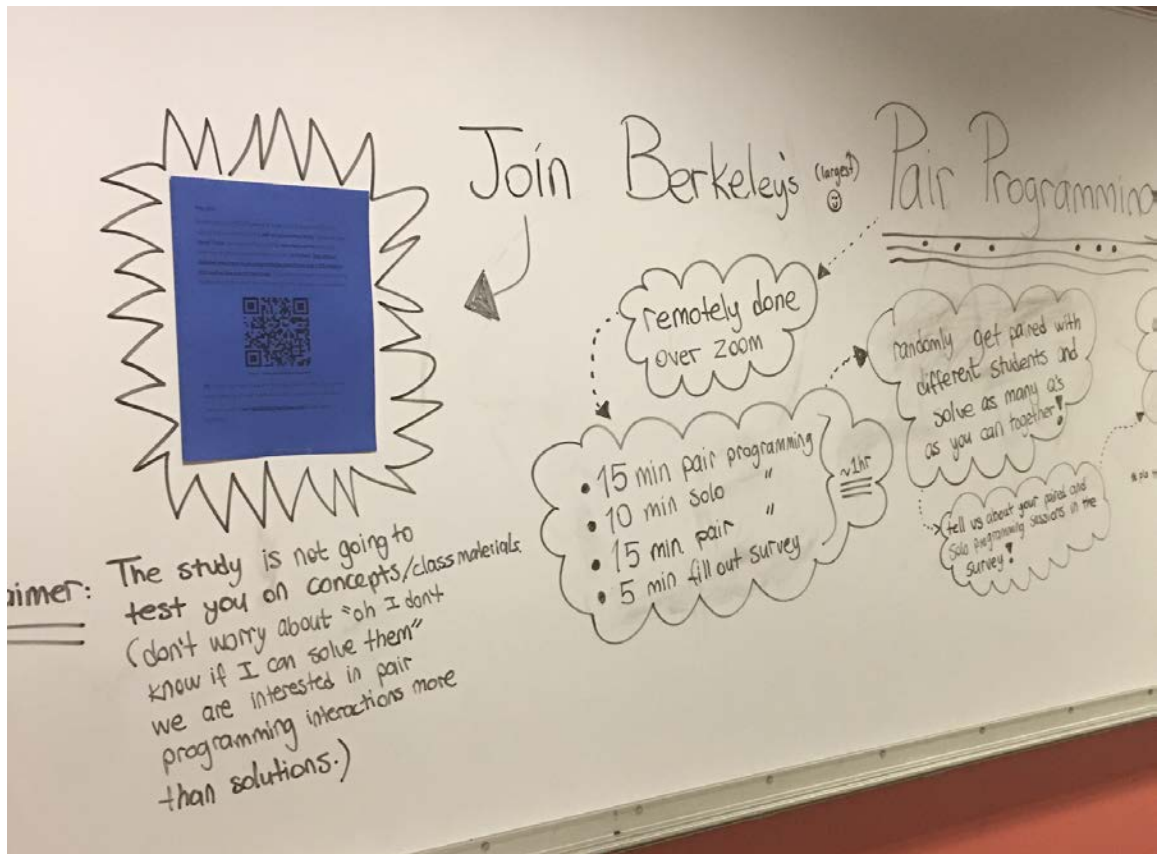


Figure 7: An example for the "friendlier" disclaimers for the study announcement.

With the moving away of one-organizer-per-participant model for hosting programming sessions, it was important to have a large enough group of participants attending the zoom call so odds of participants having any sense of whom their partner could be would be minimized. In order to manage scheduling larger groups of people, instead of asking participants their free time we switched to holding multiple big programming sessions on different days/times and asked which one(s) they can participate in. Then participants were manually scheduled based on their self-reported experience levels and self-reported gender identities to keep a balanced session. This had a huge impact on easy scheduling and we never cancelled another programming session or have an organizer fill in. Though in two cases we had an odd number of participants and we asked them to reschedule to a different session. We sent a zoom-etiquette instructions in reminder emails which requested participants to be on time, keep their videos and microphones off and ready their login codes in addition to restricting zoom messaging feature to prevent private messaging.

The pilot studies also allowed us to address a lot of feedback about the platform, to test running tests with python code correctly, add more feedback for validation tests of the autograder and update the layout for a better user experience. The feedback about the platform included “can’t view code editor, chat window and problem description at the same time”, “scrolling makes it difficult to write code and communicate with partner effectively (can’t see the chat window while coding if partner has comments)” which led the UI to be updated to its final format with 3 side-by-side panels containing the problem description and example input and outputs, the shared code editor and the chat window as shown in 8

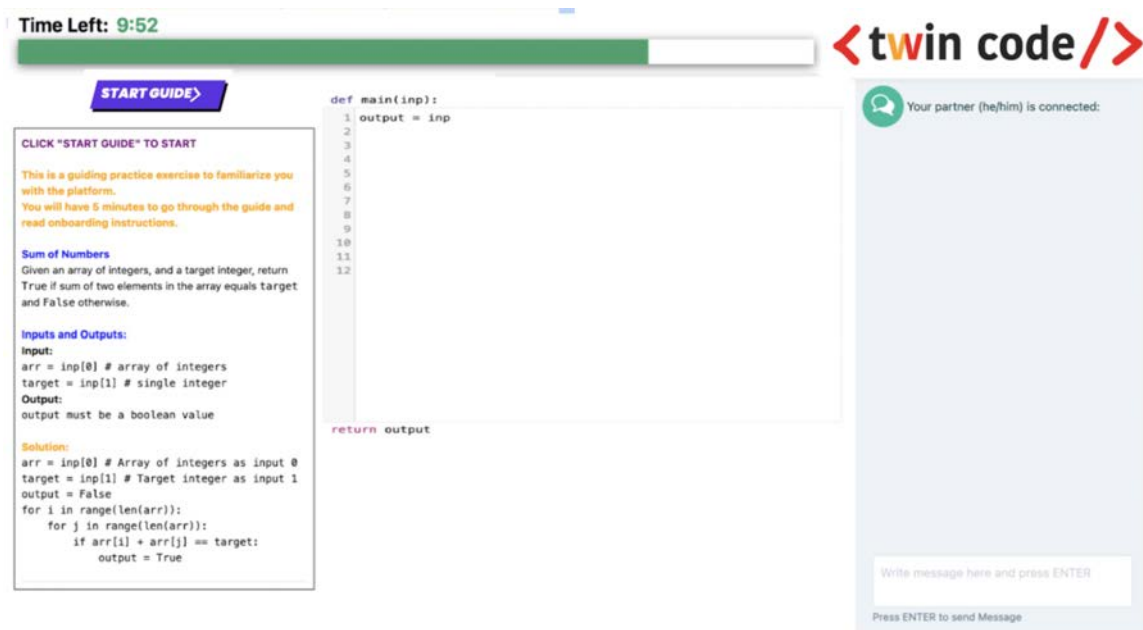


Figure 8: The twincode platform with 3 panels: problem description (left), shared code editor (middle), and chat window (right).

Finally the general feedback on running a full session helped us flash out an automated official session flow using Twincode. We addressed participants’ requests for being presented with instructions beforehand to keep the introductions

consistently short in the zoom calls, added a 5 minute walk-through practice problem into the platform before the study to help students onboard Twincode platform features as can be seen in Figure 8. We also integrated the surveys and consent forms into the Twincode platform along with timers to indicate how much time students in the programming session and to fill up each questionnaire rather than sending links individually over zoom to the participants. After these updates none of the official study sessions were went over an hour, and no student left at the end of the session without finishing up the final consent forms.

Overall, the pilot studies allowed us to get early feedback on (i) the scheduling and planning process leading up to the study (ii) the comprehensibility, internal consistency and presentation of the instructions and questionnaires, (ii) the usability and performance of the Twincode platform, and (iii) the applicability of the chat-utterance labeling [40] as shown in Figure 5.

## 4.2 Recruiting and Participants

The participants were undergraduate students enrolled in CS88 (Computational Structures in Data Science) or CS61A (The Structure and Interpretation of Computer Programs). The recruitment process included piazza/ed posts, in-person lab announcements, email blasts through student organizations such as CS scholars, whiteboard announcements and physical flyers (as seen in Figure 9) that were located around Soda Hall's indoor and outdoor student spaces such as classrooms, elevators, bathrooms and study lounges. During the flyer-posting process we also spread the information via word of mouth to many students and TAs.



### Hey, you!

We are a group of EECS graduate students and we are looking for volunteers to participate in a **pair programming study**. Study will take **about 1 hour**, during which you will be **remotely paired** with other students to solve programming questions in **Python**. **You will get detailed solutions to all programming questions and a \$15 Amazon gift card at the end of the study**. If you are interested in participating, please fill out this short recruitment form to indicate your availability:



<https://forms.gle/Qq4ZH8ZbVpcxavb47>

**PS.** Choosing to participate in this study will not affect your grades, class standing, or relationships with any instructors in any way :)

Please contact **Asli** ([asliakalin@berkeley.edu](mailto:asliakalin@berkeley.edu)) if you have any questions.

Figure 9: The physical flyers containing the recruitment call with a QR code linked to the participation form.



Figure 10: Some of the locations where flyers are posted in Soda Hall: lab rooms and Wozniak Lounge.

A total of 155 students filled out the participation form during recruitment for in the study. The distribution of where each student heard about the study is shown in Table 1.

Sources of Recruitment	Number of Students
Flyers in Soda Hall	71
CS61A lab & lecture announcements	55
CS88 lab & lecture announcements	28
Piazza (classroom question board) announcements	22
Student organization email announcements	11

Table 1: Where did students hear about the study?

In the participation interest form, students also self-reported their year, gender, experience level with python, and confirmed that they are at least 18 years old. Optionally, students also listed out any other programming languages they know and the number of upper division courses they have taken (including the ones they are currently taking). Data for all interested students can be seen in Table 11.

Student Information	Number of Students	
Year	Freshman	77 (49.68%)
	Sophomore	40 (25.81%)
	Junior	23 (14.84%)
	Senior	15 (9.67%)
Gender	Woman	78 (50.32%)
	Man	75 (48.39%)
	Non-Binary	1 (0.64%)
	Not disclosed	1 (0.64%)
Experience Level with Python	Medium	68 (43.78%)
	Low	38 (24.52%)
	High	34 (21.94%)
	Very Low	10 (6.45%)
	Very High	5 (3.23%)
Number of upper division courses*	0	47
	1	24
	2	8
	3	11
	4	14
	5	7
	6 or more	19
Any other programming languages*	Java	91
	C or C++	41
	R	33
	SQL	14
	javascript	13
	html	12
	css	10
	scheme	7
	go/golang	7
	matlab	5
	risc	3
	react	1
	snap	1

Figure 11: The table shows the distribution of students for demographic information collected in the participation form. The ones marked with \* signals that these were optional questions in the survey and not all students filled in a response, therefore no percentages are given. The experience level in python required a multiple choice answer and the rest took free text input.

Interested students submitted an availability form to voluntarily take part in the study as an interesting experience in remote pair programming, a phenomenon they were familiar with through CS88 and CS61A as the study was primarily advertised as a “pair programming study.”

Student Information	Number of Students	
Year	Freshman	27 (49.09%)
	Sophomore	17 (30.91%)
	Junior	6 (10.91%)
	Senior	5 (9.09%)
Gender	Woman	27 (49.09%)
	Man	27 (49.09%)
	"Who cares"	1 (1.82%)
Experience Level with Python	Medium	23 (41.82%)
	High	26 (29.09%)
	Low	13 (23.64%)
	Very High	2 (3.64%)
	Very Low	1 (1.82%)
Number of upper division courses*	0	14
	1	10
	2	2
	3	5
	4	6
	5	2
	6 or more	10
Any other programming languages*	Java	39
	C or C++	22
	R	5
	SQL	5
	javascript	3
	html	5
	css	4
	scheme	4
	go/golang	3
	matlab	2
	risc	2

Figure 12: The table showing the demographic information collected from the 55 participants who participated in the study (as opposed to Table 11 which included information on all interested students. This table also includes invalid participants later dropped from the study for various reasons.

Looking at all attendees (i.e who showed up and at least started the study) and comparing it to the pool of all interested participants (11, there are a few interesting changes in the pool of students. Firstly, in terms of the class levels between interested vs participating students, while the percentage of freshmen and senior students approximately remains the same, more sophomore students end up actually joining the pair programming study than juniors: sophomores consisted of the 25.81% of interested students, they consist of 30.91% of the students who attended a study session; whereas juniors were 14.84% of the interested students yet they ended up only covering 10.91% of the students who attended a session. This could be a result of increasing coursework or already existing study groups making it more difficult to attend the study as a junior, yet seniors did not show a change in their representation and consistently make up around 9% of both groups. This could be an indication that pair programming practices would be

more useful if they are introduced in Freshman or Sophomore years as Juniors might be less likely to opt-in for a new practice. In terms of gender, of all the interested students 50.32% of them were women and about 49% of students who attended. There were an equal number of men and women in the attendance pool which shows a similar distribution as the interested students pool. Finally, comparing the self-reported experience levels in Python across the interested vs attended students there is another interesting shift towards higher experience levels. Amongst interested students 6.45% identified with “very low” experience, which dropped significantly to 1.82% among students who attended a session. As for students identifying with “low” experience levels, they consist of 24.52% of all interested students and 23.64% of students who ended up attending a session, slightly decreasing. Similarly, “medium” skill level was 43.78% of interested students and declined to 41.82% of students who attended a session. On the other hand, “high” skill level identifications percentages displayed increased representation in students attended-the-study compared to students interested-in-study: 21.94% of interested students selected “high” experience with python, which increased to a 29.09% amongst the students who attended the study. Overall it looks like students with “very low” experience levels are least likely to follow through with attending a pair programming session even when they initially show interest. This could be linked to the low experienced students fearing being judged or not getting anything out of the experience due to their lack of confidence. Pair programming practices should be tailored to encourage such students who self-report low experience levels so they can benefit from positive learning outcomes of pair programming as well.

		# of women	# of men
Year	Freshman	12	15
	Sophomore	8	9
	Junior	3	3
	Senior	5	0
Experience Level	Very High	1	1
	High	8	8
	Medium	8	15
	Low	10	8
	Very Low	1	0
Number of upper division courses	0	6	8
	1	5	5
	2	2	0
	3	2	3
	4	2	4
	5	1	1
	6 or more	7	3

Figure 13: The table showing the demographic information collected from the 55 participants in 12 analyzed for each gender.

Figure 13 contains a table that separately looks at the men and women amongst the students who have attended a pair programming session. In terms of



distribution of class years, more women in higher levels attended the pair programming study such as 5 seniors compared to 0 men in senior standing. This is also reflected in the higher number of upper division courses taken by women (i.e 7 women reported 6 or more upper division courses compared to 3 men) that women could have a higher academic standing on average while still willing to attend a pair programming session. For men, on the other hand, higher the standing the fewer students attended pair programming. Interestingly despite women's higher class standing, more women self reported "Low" and "Very Low" confidence in python experience compared to men. While both gender groups had a median experience level of "medium" women experience distribution skewed towards lower confidence. This is consistent with past literature where women tend to underestimate their experience and skills, regardless of their real experience or skill levels.

Among the 55 total attendees, 52 participants, or 26 pairs fully participated in the study. Due to the platform failure in the very first session hosted (during the practice exercise), all participants had to be rescheduled to a later session to return and restart the study, but unfortunately 3 students did not return.

Amongst the 52 participants, 3 pairs were eventually dropped due to (i) a pair disclosing their identities to one another violating the instructions, (ii) one idle partner not actively participating in any paired exercises and letting one partner working alone, and (iii) one partner who accidentally exited out of Twincode during the second paired session, thus preventing metrics to be collected in the second half of the study. The final number of accepted pairs was 23, with a total of 46 valid subjects. For the valid subjects, 26 identified as a woman (56.52%) and 20 (43.48%) identified as a man. The percentage of women participants is significantly above the percentage of EECS undergraduate enrollment (23%) and L&S Computer Science enrollment (26%) in UC Berkeley according to the Fall 2022 official statistics. One student who did not identify as a man or a woman amongst the valid participants was one of the disqualified pairs, the other valid participants all self-identified as man or woman. The demographic statistics of the participants in the study is shown below. Note also that despite the 6 dropped subjects, the percentage of women in the control (12 women, 52.17%) and treatment (14 women, 60.87%) groups were close to each other. Compared to students who showed interest in the study, more students who identified as women ended up actually showed up to participate in the study.

All in all, the recruitment process lasted about a four weeks in order to recruit enough participants from CS61A and CS88. During the recruitment phase the emphasis was on the pair programming aspect of the study and no mention of gender bias or involvement of deception was made. We remarked that this study would not affect their grades or their class relationships in any way and that participants should be at least 18 years old to participate. Once recruitment phase was completed, sessions were organized based on responses from the participants in the scheduling availability participants filled out during recruiting. Out of 155 students showing interest in the study, only 79 responded to the scheduling form, all of which were invited to a study session.

# Participant Scheduling

Please mark below all the sessions you are available to attend. We will contact you (via email) with your assigned session information within a day of receiving your response.

Overview of Study:

This study is expected to take 1 hour of your time, conducted virtually over zoom. It consists of 3 sessions where you'll be solving as many problems as possible with a partner (or solo) on the pair programming platform we created (yey).

[15 minutes] Session 1: Paired session with a partner

[10 minutes] Session 2: Solo session

[15 minutes] Session 3: Paired session with a new partner

You will be compensated for your time with a \$15 Amazon gift card after the study. We will send out the gift cards via email once all sessions are done.

Figure 14: The information given to potential participants during scheduling. Potential participants then selected from a list of session dates and times given, which one(s) they could attend.

Based on the availability, 5 large sessions were organized out of the potential dates and all 79 students received invitation emails with a link to the zoom meetings, calendar invites as well as their individual login codes to the Twincode platform. We asked the students to confirm they can attend to their assigned session dates after receiving the invitation email, and sent continuous follow up reminders a week before, 3-days before and 1-day before their assigned date. Almost all students who confirmed via email or google event invite they could attend ended up attending. The reminder emails also included instructions shown in Figure 21 at the end of this section, and requested (i) participants to be on time at the beginning of the hour, (ii) ready their Twincode login codes, (iii) keep their videos and microphones off and only communicate with the hosts, and (iv) overall avoid revealing any identifiable information about themselves, their courses, location, and so on.

## 4.3 Study Workflow

Once participants gathered together on the zoom call we went over the instructions, answered any questions and made sure all participants fill out the consent form before starting the session. Unfortunately during the first big study session with over 40 participants joined in, the Twincode platform malfunctioned and we had to reschedule those participants into later sessions, some of which have not returned. Overall we hosted the remaining sessions over the course of 2 weeks with a total of 52 students attending and completing the study. When all registered students log into the Twincode platform, they were automatically allocated into control and treatment groups. Then treatment and control groups were randomly paired up to create partners by the platform and get presented programming exercises (task #1 below) to work on collaboratively. To aid with onboarding participants to the platform, we included a 5 minute exercise and a tutorial before all the actual tasks. The order of the practice exercise was not randomized unlike the remaining exercises and it was always the first exercise displayed. The internal process of Twincode is shown in Figure 15.

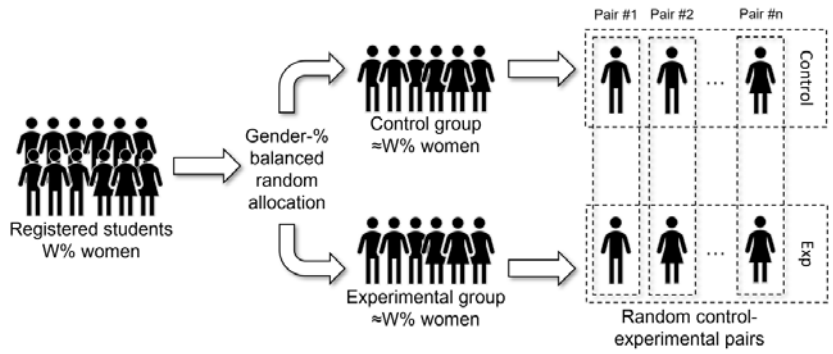


Figure 15: Random allocation into treatment and control groups made by Twincode. Twincode also balances the proportion of women in each group as much as possible. Since our study had a pretty balanced gender distribution, we can ensure that men and women were balanced in both groups.

As presented with the participant instructions the rundown of the study was as follows:

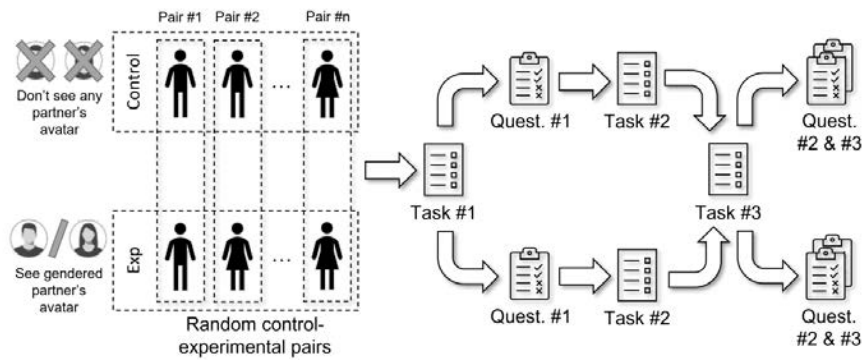


Figure 16: Diagram of the steps of the study for control and treatment groups.

1. welcoming and presenting instructions, initial consent form checking and starting the Twincode session
2. 5 minutes: tutorial and practice exercise
3. 15 minutes: pair programming 1 (Task #1)
4. 5 minutes: individual survey on paired programming 1 (Quest #1)
5. 10 minutes: solo programming (Task #2)
6. 15 minutes: paired programming 2 (Task #3)
7. 8 minutes: individual survey on pair programming 2 AND a comparative survey on both paired sessions (Quest #2 and #3)
8. signing the final consent form and exit

During each paired programming session, students were instructed to solve as many questions as possible. Once a question was successfully solved, another question would be presented to the pair until the timer runs down. These

questions were randomly selected from a pool of exercises previously vetted and they were of similar complexity. During the pair programming steps subjects in the control group had no information about the gender of their partners, but subjects in the treatment group could see gender pronouns and a gendered avatar in the chat window as previously described (Figure 4). After each pair programming step (Task #1 and #3 in 16), participants were given time to individually fill out a survey reflecting on the pair programming experience they just had including questions about perceived productivity, partner's technical competency compared to their own as well as positive and negative aspects of their partner (Quest. #1,#2,#3 in Figure 16).

To try to prevent the subject from recognizing they are working with the same partner in both pair programming steps, we include a 10 minute solo session in between (Task #2 in Figure 16). No survey is presented after the solo session and subjects directly move to the second pair programming step (Task #3 in Figure 16). Similarly, if the subject passes all the tests and solves a question before the countdown, another question would be presented from a separate pool of previously vetted solo programming questions. The main purpose of the solo session is therefore to try to get students to forget about their first partners and prevent them from recognizing them in the next paired step.

After the solo programming step, the second pair programming step proceeds under the same conditions as the first pair programming step except for the swapped gender identifications presented to the treatment group. The control group continues to have no gender information about their partners. The same pair, though unknown to the partners themselves, proceeds to collaborate on new questions until their time runs out. Twincode handles presenting the pair with new questions from the pool of paired exercises that the pair didn't see in the previous pair programming step.

Once the second pair programming step is completed, the participants complete another questionnaire (Quest #2 in Figure 16) which has the same questions as Quest #1 in Figure 16 but now asking about their second partner and second pair programming step they completed, followed by questions comparing the first and second partners (Quest #3 in Figure 16). Even though Figure 16 marks them as separate surveys in the diagram, the students were presented a single form with 2 pages, first referring only to the pair programming step they've just completed, and second comparing the two partners they had throughout the study. The last part also asks about whether the control group remembers the gendered avatars of their partners or not.

Finally, after completing the final survey, all participants are informed about the deception and the real purpose of the study, and they are given the option to withdraw their data while filling out a final consent form. None of the participants chose to withdraw.

# Welcome to our Pair Programming Study!

V4 - 2022-04-25

---

## Disclaimers:

- ★ Once on the zoom call **please keep your video and mic off**. Make sure you can hear the organizers.
- ★ Your goal is to solve given problems and pass all the autograder tests in given settings (solo or with a partner i.e “paired”). Our programming platform will provide you with **a problem description, a shared code editor and a timer**. You will also have **a messaging window** to communicate with your partner. You won’t see or hear your partner besides their code edits and the messaging window.
- ★ **You must not exchange any kind of personal information with your partner while working together**. If you do, you will not receive compensation and also risk the whole study, so please don’t.
- ★ Please make sure you have **a stable internet connection throughout the study**. Once on the coding platform **do NOT refresh your page or use arrows** of the | browser, this might disconnect you from the application and we might need to reset the whole system and reconnect all participants.

## Overview:

- This study consists of 3 parts, all in python. The outline is as follows:
  - **Session 1:** Paired session with a partner **[15 minutes]**
  - **Session 2:** Solo session **[10 minutes]**
  - **Session 3:** Paired session with a new partner **[15 minutes]**
- After each paired session, you will fill out a questionnaire.

---

**Purpose:** The purpose of this research study is to better understand distributed pair programming as it has been shown it can be a valuable instructional technique.

**Procedures:** If you agree to be in this study, you will be asked to do the following: You will be asked to participate in 3 back to back programming sessions, each of which will last no more than 15 minutes. In 2 of these sessions, you will be paired with another participant. You will be asked to work together through custom coding and chat software to solve technical interview style programming challenges to the best of your abilities. You

Figure 17: Instructions shared with the participants, page 1

will be asked to fill out a survey after the pair programming sessions. You will be asked to respond to each question, but you can decide to not participate or drop out of the study if you do not wish to answer a particular question.

**Study time & location:** Participation in this study will involve a total of 60 minutes of your time. All study procedures will take place online.

**Benefits:** We hope that the information gained from the study will help us better understand the benefits of pair programming as an educational tool. You may benefit from this study by gaining more skills for handling tech interview questions. You will also have the opportunity to practice coding with another person, a valuable skill for computer science employment. It is also possible that you will not benefit from this study.

**Confidentiality:** Your study data will be handled as confidentially as possible. If results of this study are published or presented, individual names and other personally identifiable information will not be used. As a disclaimer, the confidentiality of data transmitted online cannot be guaranteed.

To minimize the risks to confidentiality, we will do the following:

- Remind you to close your browser after completing all of the study procedures in order to protect your privacy.
- Request that you and other participants do not disclose personal information during the chats to protect the privacy of all involved.
- Personal identifiers will be removed and stored separately by assigning a coded identifier after agreeing to participate in the study.
- Only the researchers will have access to your study records.
- Your personal information may be given out if required by law.

**Future use of Study data:** The research data will be maintained for possible use in future research by myself or others. I will retain this data for up to 3 years after the study is over. The same measures described above will be taken to protect confidentiality of this study data.

---

**Action Item:** Fill out the [consent form](#) to participate in the session.

---

Figure 18: Instructions shared with the participants, page 2.

## Twincode: Our Pair Programming Platform

→ **Navigation bar** will show your login ID, and “instructions” button to get a tour of the platform.

The screenshot shows the Twincode interface. At the top is a dark navigation bar with a purple "Instructions" button, the "twincode" logo, and the text "Your Login ID: 244340". Below the navigation bar is a green "Session Countdown: 12:37" bar. The main area contains a code editor with the following code:

```
def main(inp):  
1 out = inp # replace this line with your code  
  
return out
```

Below the code editor is an orange "Run Code!" button. At the bottom is a pink "Autograder Results" panel with the following text:

```
Autograder Results:  
4 test cases: 0 tests passed. 4 tests failed. The rest errored or didn't run.  
Failed Test #1  
Failed Test #2  
Failed Test #3  
Failed Test #4
```

→ **Session Countdown** will show you how much time is left in the session. Your goal will be to solve as many questions as possible (pass all tests)

→ **Code editor** is where you will write code collaboratively. Note that the first and last lines of the function is given: your input is defined as “inp” and your goal is to assign the correct output value to “out” for the given input:

```
def main(inp):  
    #your code here  
    out = #your code here  
return out
```

→ **Run Code!** button will run the autograder and show you details about input.

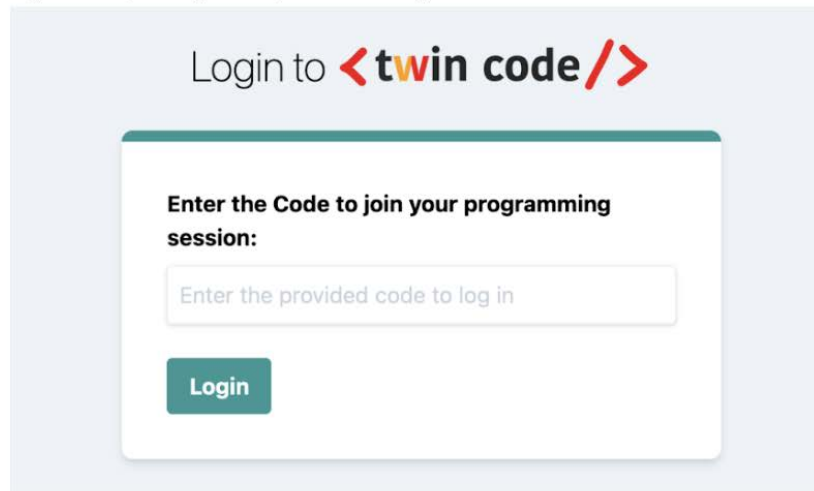
→ When all tests pass, you and your partner will automatically move to the next question. Keep in mind that partners won't see when/if the other is clicking Run Code, but if one partner

clicks “run code” passing all the tests, both you and your partner will move to the next question.

Figure 19: Instructions shared with the participants, page 3.

### Accessing the Pair Programming Platform

1. Please go to the link we will provide you on the zoom call to access the pair programming platform.
2. You will see the following screen. Please enter your **Login ID** code provided by the organizers (sent by email) and click "Login"



3. Then, you will see the following screen. Wait for the organizers to announce the **token** and click "Continue"

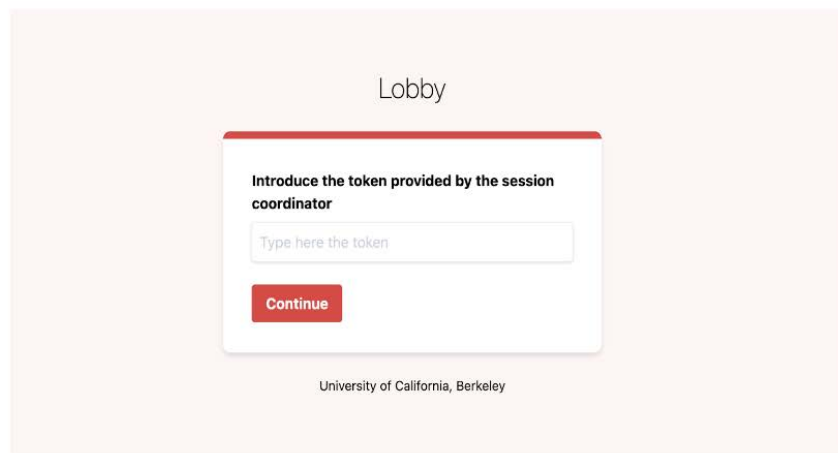
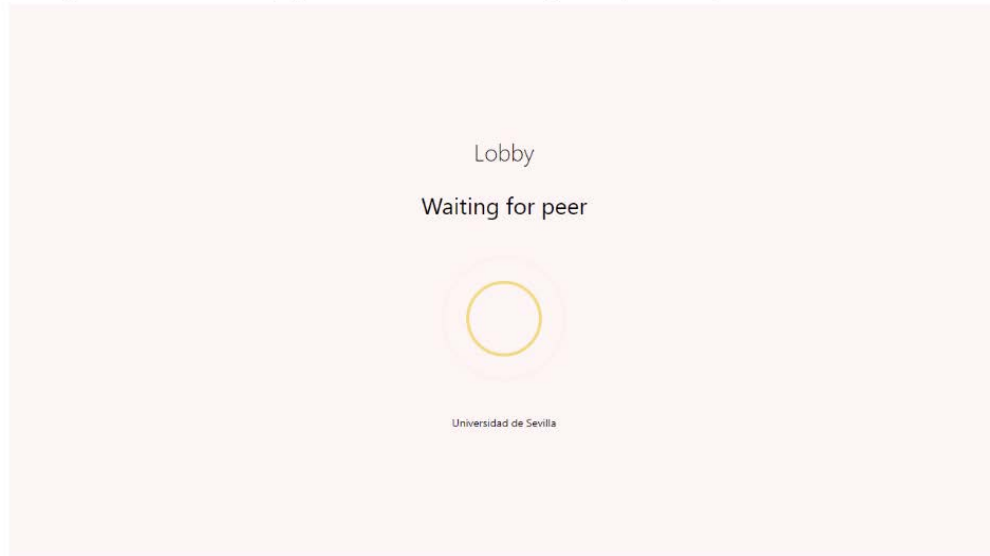


Figure 20: Instructions shared with the participants, page 4.



4. Once you are in the Lobby, you'll wait for the other participants to join.



5. Once all participants are in the Lobby, the session will start. The rest of the instructions will be displayed through the platform. Good luck and have fun!

Figure 21: Instructions shared with the participants, page 5.

## 5 Analysis: Data and Variables

The independent variables of the study are:

1. group: (treatment or control) that subjects are randomly assigned to
2. time: given in pair programming steps to the subjects for collaboratively solving the questions
3. gender: representing the subject's self-identified gender
4. ip\_gender: induced gender of partner representing the treatment group's perception, control group has no ip\_gender variable

The controlled variables of the study are:

- participants' experience levels (technical skills)
- the pool programming exercises participants are given in paired tasks (similar difficulty and in a randomly assigned order)

The depending variables of the study are grouped into 3 categories based on how they are collected:

1. questionnaires for self-reported evaluation variables
2. Twincode platform logs for coding behavior variables
3. (labeled) chat messages for communication variables

## 5.1 Perception: Self-Reported Surveys

The variables from the surveys measure subjects' perceptions about the pair programming experience and impressions of their partners. The questions are composed of Likert scales and they are computed as the average of their corresponding items. A scale of 0-10 is used in order to allow 5 to indicate a midpoint.

**pp** interval variable composed of four 0–10 numerical response items measuring the subject's own perceived productivity during each pair programming task compared to solo programming (relates to Research Question #1). Low values correspond to better solo programming productivity (i.e., "solo programming would have been more productive than pair programming"), and high values correspond to better pair programming productivity (i.e. "pair programming has been more productive than solo programming").

**pptc** interval variable composed of four 0–10 numerical response items measuring the subject's partner's perceived technical competency compared to their own after each in-pair task (relates to Research Question #2). Low values correspond to higher subject's productivity, (i.e., "I have been more productive than my partner"), and higher values correspond to higher partner's productivity (i.e. "My partner has been more productive than me"). In these responses 0 corresponds to "me"; 10 corresponds to "my partner" and 5 corresponds to "my partner and I being equal." The 4 response items are: "During the programming exercises you just did, who do you think ..."

1. ... had more knowledge and technical skills, you or your assigned partner?
2. ... has been more cooperative, you or your assigned partner?
3. ... has had a faster pace at solving the exercises, you or your assigned partner?
4. ... has led more to the solutions, you or the partner assigned to you?

**ppa** ratio variable counting the number of partner's positive aspects identified by the subject after each in-pair task (relates to Research Question #3). This variable is automatically computed from an open question item in which subjects are asked to write the most positive and negative aspects of their partners in the previously performed pair programming step. They are instructed to prefix positive aspects with a plus sign (+) and negative ones with a minus sign (-). This variable is the result of automatically counting the number of plus signs in the text of the open question "Describe your partner".

**pna** ratio variable counting the number of partner's negative aspects identified by the subject after each in-pair task (relates to Research Question #3). In a similar way to the **ppa** variable, this variable is the result of automatically counting the number of minus signs in the text of the open question "Describe your partner".

**ppgender** nominal variable measuring the perceived partner's gender during the in-pair tasks. To measure this variable, subjects are asked in questionnaire #3 whether they remember if their partners showed some avatars in chat windows or not. If the answer is no or "I don't remember" (idr), this variable is assigned the none or idr levels. If the answer is yes, then the subjects are

asked for the avatars of the first and second partner, having man, woman, or idr as options.

**cps** interval variable composed of five 0–10 numerical response items measuring whether the subject perceived better skills in their first or second partner in the in-pair tasks, (i.e., compared partners’ skills (relates to Research Question #4). Low values correspond to the first partner (i.e., “My first partner was a better partner than my second partner”), and high values correspond to the second partner (i.e. “My second partner was a better partner than my first partner”). In these questions asking about “first or second partner?” 0 corresponds to “first partner”; 10 corresponds to “second partner” and 5 corresponds to “first partner and second partner being equal.” The 5 response items are: “Comparing your partners in steps 1 and 3, who do you think ...”

1. ... provided more clear and constructive feedback, your first or second partner?
2. ... was easier to communicate with, your first or second partner?
3. ... was more knowledgeable about the subject material, your first or second partner?
4. .. would be a better project partner, your first or second partner?
5. ... would be a better Teaching Assistant, your first or second partner?

## 5.2 Behavior: Twincode Platform Logs

These variables are automatically collected by the Twincode platform and they relate to the behavior during pair programming exercises (relates to Research Question #5). In this section every variable represents a frequency, (i.e., a count), and its associated relative frequency is computed with respect to the the sum of the frequencies of the two subjects in a pair: For example, suppose that subjects  $i$  and  $j$  are the two members of a pair, and  $v_i$  and  $v_j$  are the corresponding values of the  $v$  variable. In that case, the relative frequency for each subject would be  $\frac{v_i}{v_i+v_j}$  and  $\frac{v_j}{v_i+v_j}$ , respectively.

**sca** / **sca\_rf** Ratio scale variables representing the count and relative frequency of characters added by a subject to the source code window during an in-pair task (**s**ource **c**ode **a**dditions).

**scd** / **scd\_rf** Ratio scale variables representing the count and relative frequency of characters deleted by a subject from the source code window during an in-pair task. (**s**ource **c**ode **d**eletions).

**okv** / **okv\_rf** Ratio scale variables representing the count and relative frequency of successful (**ok**) *v*alidations of the source code performed by a subject during an in-pair task.

**kov** / **kov\_rf** Ratio scale variables representing the count and relative frequency of unsuccessful (**ko**) *v*alidations of the source code performed by a subject during an in-pair task.

**dm** / **dm\_rf** Ratio scale variables representing the count and relative frequency of *d*ialog *m*essages (chat utterances) sent by a subject during an in-pair task.

### 5.3 Communication: Chat Logs

The chat utterances registered in the Twincode platform during the in-pair tasks were manually tagged according to two orthogonal dimensions described in Figure 5. First classifying each message as “formal” or “informal”, and then using the 13 tags proposed by [?] in a related work about collaboration in a remote pair programming environment similar to Twincode. For categorizing a message as “formal” we considered the way in which a university student would communicate textually to a professor; otherwise the message was tagged as “informal”. For the tagging process, we followed a process inspired by the work of [?], in which two researchers each tagged 60% of the data, covering all dialogue messages. The overlapping subset of 20%, which was used for the initial training, established the inter-coder reliability using Cohen’s *kappa*, which was  $\kappa = 0.796$  for the formal/informal tags and  $\kappa = 0.754$  for the tags in [?], both indicating *substantial* agreement and sufficient reliability for further coding according to [?].

The response variables related to the manual tagging of the chat utterances (relates to Research Question #6 and #7) correspond to the tags in Figure 5 and are listed below. Every variable represents a frequency, i.e., a count, and its associated relative frequency is computed with respect to the number of chat utterances generated by the subject during an in-pair task, which is defined by the **dm** variable specified in previous section as “dialogue messages.”

- i** / **i\_rf** Ratio scale variables representing the absolute and relative frequency of *i*nformal messages generated by a subject during an in-pair task.
- f** / **f\_rf** Ratio scale variables representing the absolute and relative frequency of *f*ormal messages generated by a subject during an in-pair task.
- s** / **s\_rf** Ratio scale variables representing the absolute and relative frequency of *s*tatement of information or explanation messages generated by a subject during an in-pair task.
- u** / **u\_rf** Ratio scale variables representing the absolute and relative frequency of opinion or indication of *u*ncertainty messages generated by a subject during an in-pair task.
- d** / **d\_rf** Ratio scale variables representing the absolute and relative frequency of explicit or *d*irect instruction messages generated by a subject during an in-pair task.
- su** / **su\_rf** Ratio scale variables representing the absolute and relative frequency of polite or indirect instruction or *s*uggestion messages generated by a subject during an in-pair task.
- ack** / **ack\_rf** Ratio scale variables representing the absolute and relative frequency of *a*cknowledgment messages generated by a subject during an in-pair task.
- m** / **m\_rf** Ratio scale variables representing the absolute and relative frequency of *m*eta-comment or reflection messages generated by a subject during an in-pair task.
- qyn** / **qyn\_rf** Ratio scale variables representing the absolute and relative frequency of *y*es/*n*o *q*uestion messages generated by a subject during an in-pair task.

- qwh** / **qwh\_rf** Ratio scale variables representing the absolute and relative frequency of *wh*- question (who, what, where, when, why, and how) messages generated by a subject during an in-pair task.
- ayn** / **ayn\_rf** Ratio scale variables representing the absolute and relative frequency of *answer* to *yes/no* question messages generated by a subject during an in-pair task.
- awh** / **awh\_rf** Ratio scale variables representing the absolute and relative frequency of *answer* to *wh*- question messages generated by a subject during an in-pair task.
- fp** / **fp\_rf** Ratio scale variables representing the absolute and relative frequency of *p*ositive task *f*eedback messages generated by a subject during an in-pair task.
- fnon** / **fnon\_rf** Ratio scale variables representing the absolute and relative frequency of *non*-positive task *f*eedback messages generated by a subject during an in-pair task.
- o** / **o\_rf** Ratio scale variables representing the absolute and relative frequency of *off*-task messages generated by a subject during an in-pair task.

## 6 Results

The data analysis was performed only for valid subjects considered defined as participants who: (i) have filled in all questionnaires and consent forms; (ii) have their metrics correctly collected by the Twincode platform; (iii) have been paired with another valid subject; and (iv) have not disclosed their identity (including personal identity and gender information) or their partner’s during the in-pair exercises; This resulted in 23 pairs, i.e. 46 valid subjects, with 3 pairs, i.e 6 subjects getting dropped because they violated the above criteria.

### 6.1 Correlation of Induced and Perceived Gender

We analyzed the correlation of the induced gender (*ip\_gender*) and perceived gender (*ppgender*) in both groups is analyzed to know whether the treatment is effectively administered to the subjects.

Induced Gender	Perceived Gender			
	man	woman	none	idr
man	<b>10</b> (43.48%)	5 (21.74%)	2 (8.70%)	6 (26.09%)
woman	5 (21.74%)	<b>9</b> (39.13%)	2 (8.70%)	7 (30.43%)
none	0 (0.00%)	1 (2.17%)	<b>30</b> (65.22%)	15 (32.61%)

Figure 22: Contingency table for induced and perceived genders.

As shown in Table 22 the percentage of subjects who were induced to think their partner as a man and remember perceiving a man avatar is around 43.5% whereas

in the case of woman avatars the percentage is 39.13%. We decided to exclude the treatment subjects for whom the induced gender did not match the perceived gender, because we considered that the treatment had not been sufficiently effective in their cases. We kept those subject in the control group who did not perceive any avatar or didn't remember it, and discarded the rest. As a result, we kept 22 subjects in the control group (10 men, 12 women) but only 9 (3 men, 6 women) in the treatment group.

## 6.2 Between-groups Analysis

For the analysis between the treatment and control groups, distance between the measurements collected from two paired steps as an absolute value is used. As per our hypothesis claiming a gender bias effect, this distance should be smaller for the control group since these subjects have no information about their partner's genders.

For every response variable except for **cps** one-tailed mean difference test is performed between the groups and a t-test or a Mann-Whitney U test depending on the results of the normality assumption tests. For the **cps** variable, we expected the mean to be around the middle point (i.e 5) for the control group between partners since they would unknowingly be comparing the skills of the same person. For the treatment group, we expected the mean to be skewed towards 0 (i.e perceiving man partner as more skilful) or 10 (i.e perceiving woman partner as more skilful).

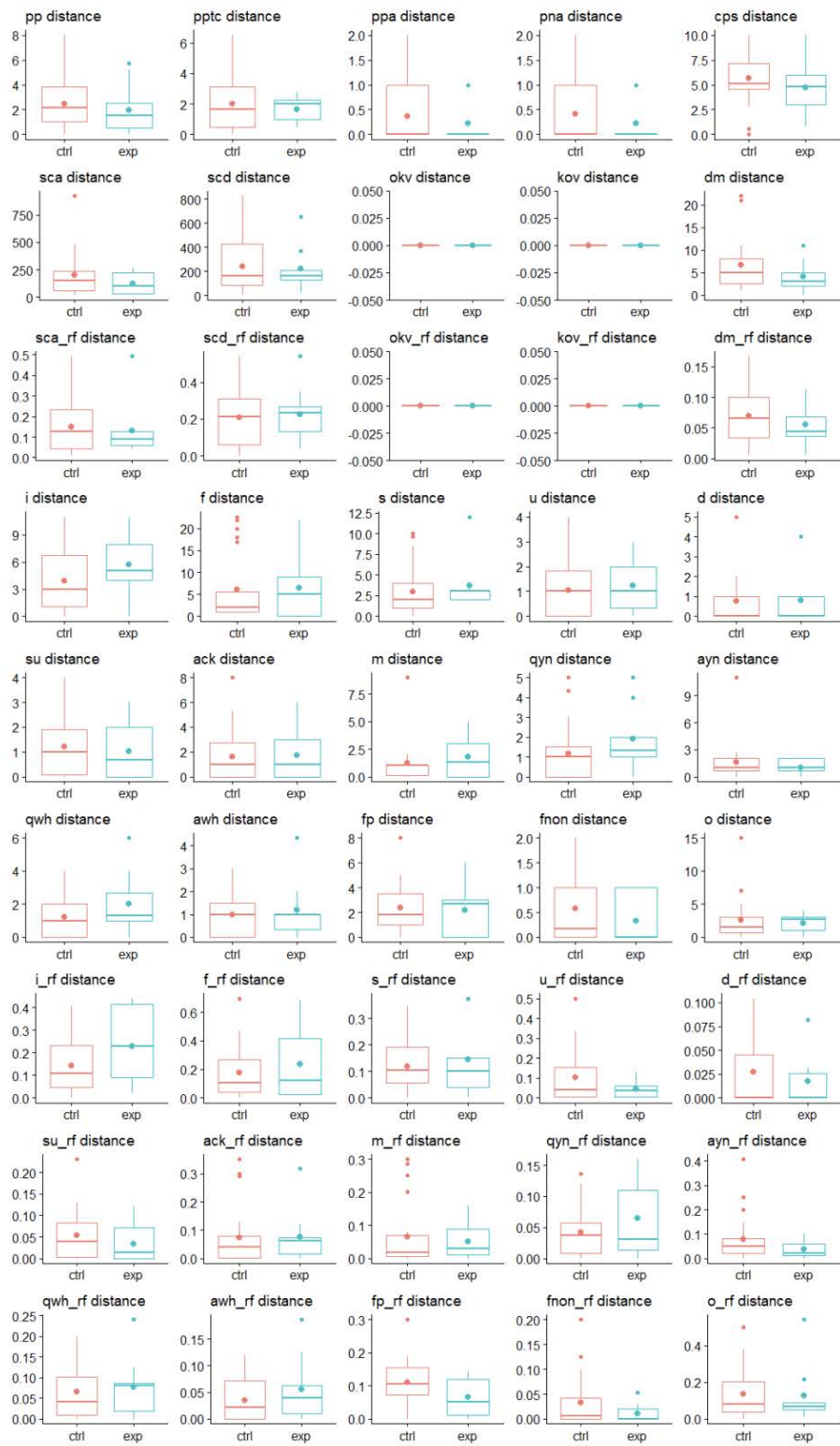


Figure 23: Boxplots of the 45 independent variables for between-groups analysis. No significant differences are observed between treatment and control groups for any of the variables.

Contrary to our research hypothesis, no significant differences were observed for

( $\alpha = 0.05$ ) between the treatment and control groups for any of the independent variables. Figure 23 shows the boxplots visualizing the means for each group indicate that the differences are very small.

### 6.3 Within-groups Analysis

We analyzed whether there were differences between independent variables when the same subjects perceived their partners as men or women according to our research hypothesis. We performed a two-sided paired mean difference test for every response variable except for **cps** using the perceived gender (**ppgender**) as a within-subjects variable, and applying a t-test or a Wilcoxon test depending on the results of the normality assumption tests.



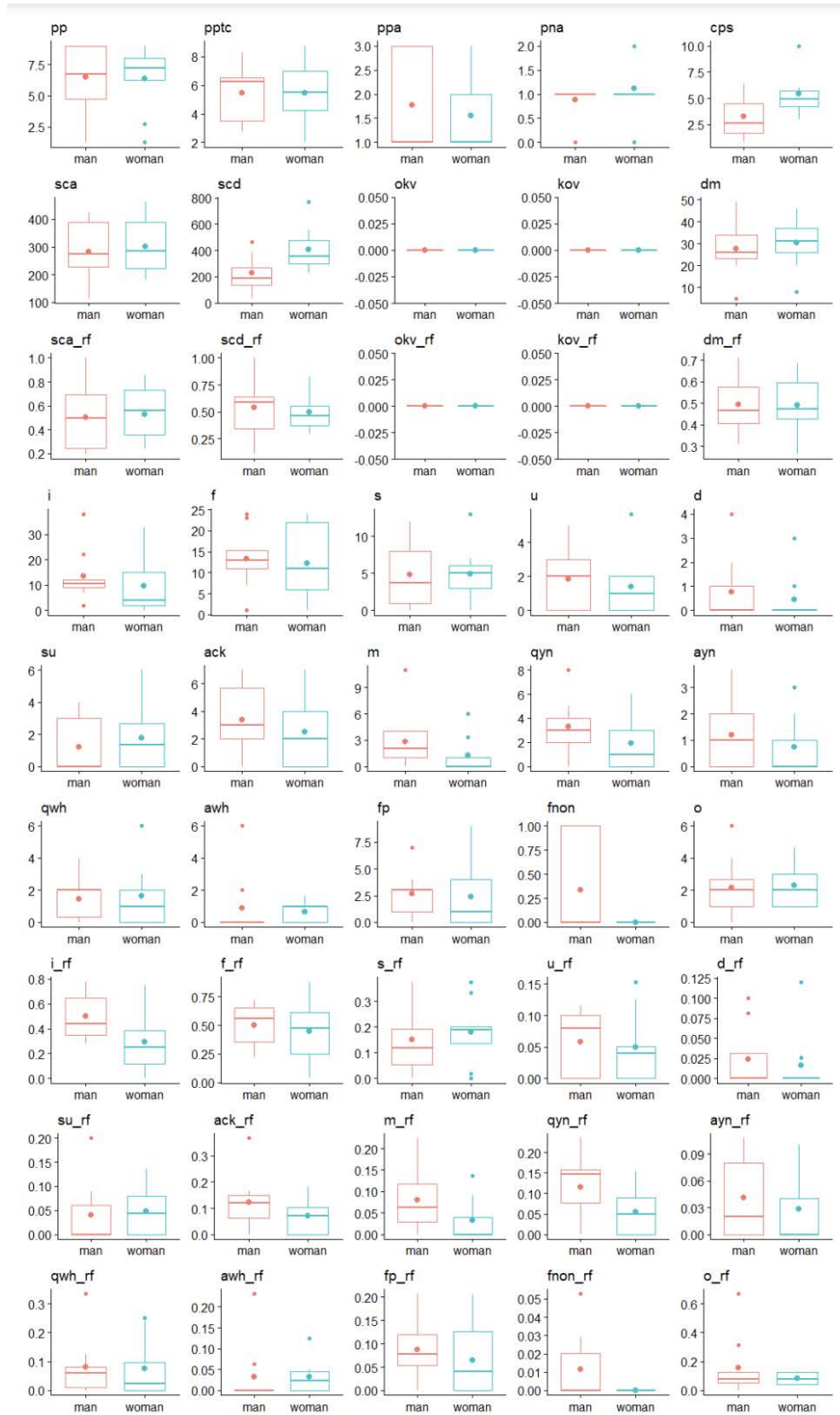


Figure 24: Boxplots of the 45 independent variables for between-groups analysis. Significant differences are observed in 4 variables: source code deletions, relative frequency of informal messages, relative frequency of informal messages, relative frequency of meta-comments or reflections and relative frequency of yes/no questions.

For the within-groups analysis, we also wanted to study the interaction between perceived gender of the partner and the subject’s gender. We performed the corresponding mixed-model two-way ANOVAs with the perceived gender (ppgender) as a within-subjects variable and the subject’s gender (gender) as a between-subjects variable. We find statistically significant differences at  $\alpha=0.05$  in the following four response variables when using the perceived partner’s gender (ppgender) as a within-subjects variable. As depicted in Figure 24 the corresponding box blots show differences between means when partners are perceived as men or women in the treatment group. Out of all independent variables, four statistically significant differences are measured where these variables passed the Shapiro-Wilk normality test and were analyzed using two-sided paired t-test. Their effect sizes were computed using Cohen’s  $d$ .

- **source code deletions (scd)**: the test detected that subjects deleted more source characters when they perceived their partners as a woman, with a *moderate* effect size ( $d = -0.775$ ) and ( $p = 0.0485$ )
- **relative frequency of informal messages (i\_rf)**: the test detected that subjects increased the relative frequency of informal messages when they perceived their partners as a man, with a *large* effect size ( $d = 1.050$ ) and ( $p = 0.0138$ ).
- **relative frequency of meta-comments or reflections(m\_rf)**: the test detected that subjects increased the relative frequency of meta-comments or reflections when they perceived their partners as a man, with a *large* effect size ( $d = 0.829$ ) and ( $p = 0.0377$ ).
- **relative frequency of yes/no questions (qyn\_rf)**: the test detected that subjects increased the relative frequency of yes/no questions when they perceived their partners as a man, with a *large* effect size ( $d = 0.880$ ) and ( $p = 0.0297$ ) .

#### 6.4 Discussion, Limitations, Threats to Validity

In this study we only observed statistically significant effects within the treatment group (when comparing how subjects acted when they perceived their partner as a man and when they perceived their partner as a woman) in four of the 45 dependent variables. Among these variables, one was related to the changes in behavior (source code deletions) and the other three were related with communication (informal vs formal messages, meta-comments/reflections, yes/no questions in chat utterances). We found that when perceiving one’s partner as a woman, subjects tended to delete more source code and used a lower relative frequency of informal messages, reflections, and yes/no questions.

We also observed a low effectiveness of the treatment factor. Operationalizing implicit gender bias into a treatment is not a straightforward task, and given the results in Figure 22, we may not have designed our treatment as adequately as we intended, thus threatening construct validity.

While designing the treatment we wanted to avoid focusing too much of the subject’s attention on gender to avoid suspicion or awareness of being observed about that fact. We thought that explicitly letting students take note of the gender of their partner as one of the observed factors could lead them to behave unnaturally during the collaboration or accidentally expose gender information during communications, thus invalidating the study.

Compared to the Seville study in which visually less-distinctive (smaller, black and white, silhouette-based) avatars and no pronouns were used with an effectiveness close to 60%, ours seems to have a lower treatment effectiveness of 40% despite having used both gendered avatars and pronouns. This decrease in treatment compared to the Seville study could have been affected by a few factors.

Firstly, in order to remove suspicion from gender and encourage participation without students worrying about their skill levels, we placed a heavy emphasis on “collaboration” and “pair interaction” while introducing the study and giving instructions. This may have inadvertently caused students to not mind at all to the induced gender information, assuming it has nothing to do with the task at hand and only focus on working together, regardless of whom they may be paired with. One of the participants responding “who cares” in the participant interest form could be a signal in this direction.

Additionally, our study was hosted in a fully remote setting which increases the likelihood of potential distractions compared to a tightly controlled laboratory setting. Our experience with a few idle participants who joined the Zoom call, followed the instructions, but then did not contribute any code or communication (going completely idle once the study begins) signal that potential distractions could also play a part.

Each pair programming session is only 15 minutes long, so students might not have had enough time to commit the induced gender of their partner to memory long enough to recall it correctly at the very end of the study. Another possible external factor is the so-called “Zoom burnout” [41] describing the fatigue and exhaustion caused by prolonged use of video conferencing during the pandemic, which may have influenced the attention span, attention direction or performance of participants. Since Zoom had been the default classroom setting for almost two years at the time of our study, the students could also be selective in the information they recalled about their partners. Having she/her pronouns in one’s Zoom name and using various Zoom profile pictures have become standard practices. Therefore students might not have paid attention to these bits of information to recall them once the interaction was over. Perhaps asking the gender of their partner as an open text question in each survey could have remedied this issue.

As a result of the low contingency rate, the remaining accepted participants represent a small number of selected subjects, especially in the treatment group ( $n=9$ ) so these results must be considered carefully. We can also note that unlike majority of the studies studying pair programming, number of women in both control and treatment groups represent the majority (with 6 out of the 9 people in treatment group and 12 out of the 22 subjects in the control group identifying as women) so these findings can shed light on interesting insights.

Overall the small sample size of the study and the low effectiveness of the treatment group propose a clear threat to conclusion validity that could only be mitigated by taking the outcomes as provisional and running more replications with bigger samples.

## 7 Future Work

This study explored the existence and extent of implicit gender bias in pair programming interactions, finding evidence that implicit gender bias could affect one’s coding and communication behaviors of pairs depending on the perceived gender of an one’s partner. Building on our findings, future work could run the study in 2 phases, going beyond identifying the gender bias within-subjects, but attempting to counter it with various interventions.

The first phase helps us determine the dimensions of gender bias, and specific behaviors that arise when the perception of gender changes as we attempted in this study with a small sample size; and the second phase would allow us to conduct a similar follow-up study with interventions that are designed to reduce biased behaviors, so we can produce counter-gender-biased behaviors, increase the productivity and efficiency of pair programming for all participants. Based on the findings of this study, potential interventions could be (i) tweaking the platform to enforce driver and navigator roles and only allowing the driver to edit code, thus preventing excessive source code deletions, (ii) using a visualization of “skill” to partners and to increment women’s self-reported skill levels as a way to counteract self-underestimation and (ii) implementing another onboarding step for smoother pair communications (just like the practice problem we decided to include after the pilot studies, which taught us that subjects weren’t readily comfortable getting to work on an newly introduced platform, future work could include a step for kick-starting better pair interactions such as an icebreaker to get the pair talking informally before the session so subjects can comfortably ask “easy” yes/no questions without fear of judgement and make meta comments like “hmm”) Research shows that identifying and reducing such bias and the gender gap is not only better for companies [1], but also that there are indeed tangible things that can be done to achieve it [44, 19]. Tuning pair-programming to detect and reduce implicit gender bias during that task could help and we hope future studies will consider a variety of interventions to measure their effectiveness in countering implicit gender bias.

## References

- [1] Why diversity and inclusion matter (quick take), Feb 2023.
- [2] K. Beck. *Extreme Programming Explained: Embrace Change*. An Alan R. Apt Book Series. Addison-Wesley, 2000.
- [3] A. Begel and N. Nagappan. Pair programming: what’s in it for me? In *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, pages 120–128, 2008.
- [4] J. Bennedsen and M. E. Caspersen. Failure rates in introductory programming. *SIGCSE Bull.*, 39(2):32–36, jun 2007.
- [5] T. Bipp, A. Lepper, and D. Schmedding. Pair programming in software development teams—an empirical study of its benefits. *Information and Software Technology*, 50(3):231–240, 2008.
- [6] G. Braught, T. Wahls, and L. M. Eby. The case for pair programming in the computer science classroom. *ACM Transactions on Computing Education (TOCE)*, 11(1):1–21, 2011.

- [7] J. C. Carver, L. Henderson, L. He, J. Hodges, and D. Reese. Increased retention of early computer science and software engineering students using pair programming. In *20th Conference on Software Engineering Education & Training (CSEET'07)*, pages 115–122. IEEE, 2007.
- [8] J. Chao and G. Atli. Critical personality traits in successful pair programming. In *AGILE 2006 (AGILE'06)*, pages 5–pp. IEEE, 2006.
- [9] E. A. Chaparro, A. Yuksel, P. Romero, and S. Bryant. Factors affecting the perceived effectiveness of pair programming in higher education. In *Proceedings of the 17th Workshop of the Psychology of Programming Interest Group*, 2005.
- [10] K. S. Choi. Evaluating gender significance within a pair programming context. In *Proceedings of the Hawaii International Conference on System Sciences*, pages 4817–4825, 2013.
- [11] K. S. Choi. A comparative analysis of different gender pair combinations in pair programming. *Behaviour & Information Technology*, 34(8):825–837, 2015.
- [12] K. S. Choi. A comparative analysis of different gender pair combinations in pair programming. *Behaviour & Information Technology*, 34(8):825–837, 2015.
- [13] K. S. Choi, F. P. Deek, and I. Im. Exploring the underlying aspects of pair programming: The impact of personality. *Information and Software Technology*, 50(11):1114–1126, 2008.
- [14] L. Cohen, L. Manion, and K. Morrison. *Research Methods in Education*. Routledge, 8th edition, 2018.
- [15] B. J. da Silva Estácio and R. Prikladnicki. Distributed pair programming: A systematic literature review. *Information and Software Technology*, 63:1–10, 2015.
- [16] T. H. DeClue. Pair programming and pair trading: effects on learning and motivation in a cs2 course. *Journal of Computing Sciences in colleges*, 18(5):49–56, 2003.
- [17] N. K. Denzin. *Sociological Methods: A Sourcebook*. Aldine Transaction, 5th edition, 2006.
- [18] A. Durán, P. Fernández, B. Bernárdez, N. Weinman, A. Akalın, and A. Fox. Exploring gender bias in remote pair programming among software engineering students: The twincode original study and first external replication, 2023.
- [19] D. Eckles, R. F. Kizilcec, and E. Bakshy. Estimating peer effects in networks with peer encouragement designs. *Proceedings of the National Academy of Sciences*, 113(27):7316–7322, 2016.
- [20] K. El-Refai, D. Kwon, D. Brincau, A. Akalın, A. Fox, P. F. Montes, and A. D. Toro. Twincode: An instrumented platform for pair programming research. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 2*, page 1264, 2023.
- [21] O. Gómez, M. Solari, C. Calvache, and A. Ledezma-Carrizalez. A controlled experiment on productivity of pair programming gender combinations: Preliminary results. In *Proceedings of the XX Ibero-American Conference on Software Engineering*, pages 197–210, 05 2017.

- [22] B. Hanks. Empirical studies of distributed pair programming. *Doctoral dissertation*, 2005.
- [23] B. Hanks. Student attitudes toward pair programming. *SIGCSE Bull.*, 38(3):113–117, jun 2006.
- [24] B. Hanks, S. Fitzgerald, R. McCauley, L. Murphy, and C. Zander. Pair programming in education: a literature review. *Computer Science Education*, 21(2):135–173, 2011.
- [25] B. Hanks, C. McDowell, D. Draper, and M. Krnjajic. Program quality with pair programming in cs1. In *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education*, pages 176–180, 2004.
- [26] S. I. Hofer. Studying gender bias in physics grading: The role of teaching experience and country. *International Journal of Science Education*, 37(17):2879–2905, 2015.
- [27] N. Jacobson and S. K. Schaefer. Pair programming in cs1: Overcoming objections to its adoption. *SIGCSE Bull.*, 40(2):93–96, jun 2008.
- [28] L. Jarratt, N. A. Bowman, K. C. Culver, and A. M. Segre. A large-scale experimental study of gender and pair composition in pair programming. In *Proceedings of the ACM Conference on Innovation and Technology in Computer Science Education*, pages 176–181, 2019.
- [29] R. Jensen. A pair programming experience. 2003.
- [30] N. Katira, L. Williams, and J. Osborne. Towards increasing the compatibility of student pair programmers. In *International Conference on Software Engineering*, pages 625–626, 2005.
- [31] S. Kaur Kuttal, K. Gerstner, and A. Bejarano. Remote pair programming in online cs education: Investigating through a gender lens. In *2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 75–85, Oct 2019.
- [32] J. Margolis and A. Fisher. *Unlocking the clubhouse: Women in computing*. MIT press, 2002.
- [33] R. F. Martell, D. M. Lane, and C. Emrich. Male-female differences: A computer simulation. *American Psychologist*, 51(2):157–158, 7 1996.
- [34] C. McDowell, L. Werner, H. E. Bullock, and J. Fernald. The impact of pair programming on student performance, perception and persistence. In *25th International Conference on Software Engineering, 2003. Proceedings.*, pages 602–607. IEEE, 2003.
- [35] C. McDowell, L. Werner, H. E. Bullock, and J. Fernald. Pair programming improves student retention, confidence, and program quality. *Commun. ACM*, 49(8):90–95, aug 2006.
- [36] M. M. Muller and W. F. Tichy. Case study: extreme programming in a university environment. In *Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001*, pages 537–544. IEEE, 2001.

- [37] N. Nagappan, L. Williams, E. Wiebe, C. Miller, S. Balik, M. Ferzli, and J. Petlick. Pair learning: With an eye toward future success. In F. Maurer and D. Wells, editors, *Extreme Programming and Agile Methods - XP/Agile Universe 2003*, pages 185–198, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.
- [38] C. O’Connor and H. Joffe. Intercoder reliability in qualitative research: Debates and practical guidelines. *International Journal of Qualitative Methods*, 19:1–13, 2020.
- [39] M. Phongpaibul and B. Boehm. An empirical comparison between pair development and software inspection in thailand. In *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*, pages 85–94, 2006.
- [40] F. J. Rodríguez, K. M. Price, and K. E. Boyer. Exploring the pair programming process: Characteristics of effective collaboration. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, SIGCSE ’17, page 507–512, New York, NY, USA, 2017. Association for Computing Machinery.
- [41] O. Samara and A. Monzon. Zoom burnout amidst a pandemic: Perspective from a medical student and learner. *Therapeutic Advances in Infectious Disease*, 8, 2021.
- [42] M. Syed and S. C. Nelson. Guidelines for establishing reliability when coding narrative data. *Emerging Adulthood*, 3(6):375–387, 2015.
- [43] L. Thomas, M. Ratcliffe, and A. Robertson. Code warriors and code-a-phobes: A study in attitude and pair programming. *SIGCSE Bull.*, 35(1):363–367, Jan. 2003.
- [44] M. Tomai, M. Mebane, V. Rosa, and M. Benedetti. Can computer supported collaborative learning (cscl) promote counter-stereotypical gender communication styles in male and female university students? *Procedia - Social and Behavioral Sciences*, 116:4384–4392, 2014. 5th World Conference on Educational Sciences.
- [45] T. VanDeGrift. Coupling pair programming and writing: learning about students’ perceptions and processes. In *Proceedings of the 35th SIGCSE technical symposium on Computer science education*, pages 2–6, 2004.
- [46] L. L. Werner, B. Hanks, and C. McDowell. Pair-programming helps female computer science students. *J. Educ. Resour. Comput.*, 4(1):4–es, mar 2004.
- [47] M. E. Whiting, A. Blaising, C. Barreau, L. Fiuza, N. Marda, M. Valentine, and M. S. Bernstein. Did it have to end this way? understanding the consistency of team fracture. *Proc. ACM Hum.-Comput. Interact.*, 3(CSCW), nov 2019.
- [48] L. A. Williams and R. R. Kessler. All i really need to know about pair programming i learned in kindergarten. *Communications of the ACM*, 43(5):108–114, 2000.

- [49] K. M. Ying, A. C. Martin, F. J. Rodríguez, and K. E. Boyer. Cs1 students' perspectives on the computer science gender gap: Achieving equity requires awareness. In *2021 Conference on Research in Equitable and Sustained Participation in Engineering, Computing, and Technology (RESPECT)*, pages 1–9. IEEE, 2021.
- [50] K. M. Ying, F. J. Rodríguez, A. L. Dibble, and K. E. Boyer. Understanding women's remote collaborative programming experiences: The relationship between dialogue features and reported perceptions. *Proceedings of the ACM on Human-Computer Interaction*, 4(CSCW3):1–29, 2021.