# Trade-offs for Substituting a Human with an Agent in a Pair Programming Context: The Good, the Bad, and the Ugly

Sandeep Kaur Kuttal
sandeep-kuttal@utulsa.edu
The University of Tulsa
Tulsa, Oklahoma

Bali Ong*
bao2656@utulsa.edu
The University of Tulsa
Tulsa, Oklahoma

Katherine Kwasny*
ksk2722@utulsa.edu
The University of Tulsa
Tulsa, Oklahoma

Peter Robe
pjr144@utulsa.edu
The University of Tulsa
Tulsa, Oklahoma

## ABSTRACT

Pair programming has a documented history of benefits, such as increased code quality, productivity, self-efficacy, knowledge transfer, and reduced gender gap. Research uncovered problems with pair programming related to scheduling, collocating, role imbalance, and power dynamics. We investigated the trade-offs of substituting a human with an agent to simultaneously provide benefits and alleviate obstacles in pair programming. We conducted gender-balanced studies with human-human pairs in a remote lab with 18 programmers and Wizard-of-Oz studies with 14 programmers, then analyzed results quantitatively and qualitatively. Our comparative analysis of the two studies showed no significant differences in productivity, code quality, and self-efficacy. Further, agents facilitated knowledge transfer; however, unlike humans, agents were unable to provide logical explanations or discussions. Human partners trusted and showed humility towards agents. Our results demonstrate that agents can act as effective pair programming partners and open the way towards new research on conversational agents for programming.

## CCS CONCEPTS

• **Human-centered computing → User studies**.

## KEYWORDS

Pair programming, conversational agents, avatars, empirical evaluation

**ACM Reference Format:**
Sandeep Kaur Kuttal, Bali Ong, Katherine Kwasny, and Peter Robe. 2021. Trade-offs for Substituting a Human with an Agent in a Pair Programming Context: The Good, the Bad, and the Ugly. In *CHI Conference on Human Factors in Computing Systems (CHI '21), May 8–13, 2021, Yokohama, Japan.* ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/3411764.3445659


*CHI '21, May 8–13, 2021, Yokohama, Japan*

ACM ISBN 978-1-4503-8096-6/21/05…$15.00
https://doi.org/10.1145/3411764.3445659

## 1 INTRODUCTION

Pair programming is based on the tenet "two heads are better than one" [189]. Two programmers solve a problem together, either collocated or remotely. One partner is the driver who actively creates code and controls the keyboard and mouse; the other partner is the navigator who constantly reviews the driver's work, proposes suggestions, and poses clarifying questions [129, 180, 183]. This paradigm fosters active learning, where knowledge is gained by doing, in comparison to the formal tutor-tutee relationships in intelligent tutoring systems. The benefits of pair programming are well established and include increased code quality, efficiency, knowledge transfer, and self-efficacy, as well as a reduced gender gap [15, 43, 47, 86, 140, 197]. Pair programming also helps increase retention rates [131], even for non-computer science majors [126].

However, researchers have also reported limitations with pair programming, such as difficulties scheduling and collocating pairs [68], student resistance to pairing [179, 185], and dependencies on their peers' programming abilities [56]. Furthermore, establishing a synergistic collaboration between pairs can be challenging as they might be affected by their relational interactions [87, 166] or may be predisposed to negative stereotypes [156]. These issues are often exacerbated in remote collaboration and learning.

To address these issues, we conjecture that replacing one human member of the pair with a conversational agent can help leverage the strengths provided by each member in a pair programming scenario. Agents, such as Apple's Siri [7], Google Assistant [63], and Amazon Alexa [6] have positively transformed our day-to-day lives. We conjectured that a pair programming conversational agent can bring their benefits into the programming domain.

The goal of this research is to investigate the feasibility of such a conversational agent as a partner in a pair programming context. Towards this end, we formulated the following research questions:

- RQ1: Can we continue the benefits of pair programming by replacing a human programmer with an agent?
- RQ2: What kind of knowledge is transferred between human-human and human-agent pairs?
- RQ3: Do human programmers consider the agent as their partner?

To explore these questions, we conducted a human-human study with 18 university students in a remote lab setting with gender-balanced pairs (3 man-man, 3 woman-woman, and 3 man-woman).

This study was followed by a Wizard of Oz study with 8 university students (4 men and 4 women) and 6 professional programmers (3 men and 3 women). We used quantitative and qualitative analysis to measure the similarities and differences between the human-human and human-agent studies.

## 2 BACKGROUND AND RELATED WORK ON PAIR PROGRAMMING

Pair programming is a practice in which two programmers collaborate on the analysis, design, programming, and testing of software. One member of the pair is the designated driver, who actively creates code and controls the keyboard and mouse; their partner is the navigator, who constantly reviews the keyed data to identify tactical and strategic deficiencies. These include erroneous syntax and logic, misspellings, and implementations that do not map to the design. The two of them switch roles when necessary. [180, 186]. Collaboration is an effective tool for teaching introductory programming. Students who code in pairs produce better programs, complete courses at higher rates, and perform equally on final exams when compared to students who have programmed independently [112].

Remote pair programming, also known as distributed pair programming, is comparable to collocated pair programming in regards to student performance, code quality, and productivity [167]. The COVID-19 global pandemic has limited the ability to collocate, forcing programmers to shift towards remote pair programming. Fortunately, remote pair programming is still a viable option - research has found that remote pair programming has high satisfaction rates among students [167].

Pairing has proven to be a more effective learning and motivational technique than individual programming [30, 112, 127]. Research has shown that pair programming improves design and code quality, reduces errors, and strengthens communication among teams [50, 66, 98, 113, 114, 122, 125, 184, 186]. In an educational setting, pair programming enables students to understand concepts and organize their own knowledge as a result of interactions and alternating roles [175]. Furthermore, pair programming can help teach professionals the specifics of a project and increase overall productivity [198].

Pair programming also improves the self-efficacy (confidence in the ability to perform a particular task) of the participants involved [125]. Programmers with high self-efficacy tend to be more persistent and flexible in their problem solving [13]. They are also more likely to perform better while programming [113] and are even happier [181]. Additionally, pair programming reduces the gender gap by increasing female students' confidence in programming and encouraging them to continue pursuing computer science [141, 175].

Although research has a shown positive correlation between pair programming and a reduced gender gap in computer science, women have still raised concerns about partnering with men. As Ying et al. [191] found, women were more likely to feel insecure about their role when pair programming with men. On the other hand, men were likely to see themselves as more competent or prepared. Although pairs with a woman were more likely to finish an assignment in class, women continued to rate their confidence lower, particularly when paired with a man [79]. Currently, only 18% of computer science undergraduate students are women, and this number is reduced to 10% in the workplace [158]. This imbalance makes it difficult for women to find a pair programming partner they feel comfortable with.

The positive effects of pair programming on retention and knowledge transfer enable programmers to apply learned knowledge to new programs [27, 28, 37, 104, 105, 113, 129, 131]. Studies show that pairs learned topics better [181], narrowed their gaps in knowledge [8, 36], exchanged project-related and general knowledge [132], and discovered new tools [121]. Pairs also exhibited little indication that interruptions from their partner disrupted their flow [98].

The most important part of pair programming is not the individual skill or personality of the programmers, but rather how they work together [198]. Pair programming (collocated) measured higher in both learning gains and student satisfaction compared to alternative collaboration methods, demonstrating the importance of working on shared code and collaboration [29]. Pair programming has also benefited students in other domains, like data science and engineering [142].

Despite the significant positive impact of pair programming, implementation challenges arise due to difficulties in scheduling and collocating pairs [68]. Further complicating matters, some individuals desire to work alone because they do not want to be "slowed down" by their partner [182]. In addition, successfully completing a task using this process depends on the pairs' programming abilities [55]. We conjecture that introducing an agent can eliminate these challenges, because programmers would not depend on their colleagues' schedules, locations, and abilities. Our study will enrich the understanding of the pair programming process involving a human-agent pair, as well as the feasibility of introducing such an agent for pair programming.

## 3 HUMAN-HUMAN STUDY

First, we conducted a human-human study in order to analyze the interactions between two humans in a pair programming environment. The details of the study were as follows:

### 3.1 Participants

We recruited 18 computer science students on first-come-first-serve basis. All participants had some level of experience with the Java programming language. Table 1 details the general demographics of our study participants. As an incentive, participants were given a $20 Amazon gift card.

We placed the participants into nine gender balanced pairs - 3 man-man, 3 woman-woman, and 3 man-woman. Based on the background questionnaire, we identified only binary genders (men and women) according to the self-identification of each participant's gender [26, 176]. Gender balanced data is crucial for (1) discovering gender biases in the interface designs [22, 33, 116, 149], (2) avoiding agents that support ideologies in machine learning that are unfavorable towards women [100], (3) supporting gender specific problem solving [8, 52], communication techniques, and leadership styles [87]. Also note that various factors such as ethnicity, race, and sexual orientation can impact how programmers interact with each other. Although there are various minority groups in computer

**Table 1: General Demographics of Participants in Human-Human Study**

| Pair# | Gender | Level | Age | Experience | |
|---|---|---|---|---|---|
| | | | | Prog. | Pair-Prog. |
| HH1-M1 | Man | Junior | 19-23 | 2 Years | Yes |
| HH1-M2 | Man | Sophomore | 19-23 | 2 Years | Yes |
| HH2-M3 | Man | Senior | 19-23 | >4 Years | No |
| HH2-M4 | Man | Freshman | 19-23 | 3 Years | No |
| HH3-M5 | Man | Freshman | 19-23 | 4 Years | No |
| HH3-W1 | Woman | Senior | 19-23 | 3 Years | No |
| HH4-W2 | Woman | Junior | 19-23 | 2 Years | No |
| HH4-W3 | Woman | Senior | 19-23 | 2 Years | Yes |
| HH5-W4 | Woman | Masters | 30-40 | 2 Years | No |
| HH5-W5 | Woman | Masters | 19-23 | 2 Years | No |
| HH6-M6 | Man | Senior | 19-23 | 4 Years | No |
| HH6-W6 | Woman | Senior | 19-23 | 3 Years | Yes |
| HH7-W7 | Woman | Senior | 19-23 | 2 Years | No |
| HH7-W8 | Woman | Junior | 19-23 | 2 Years | Yes |
| HH8-M7 | Man | Senior | 19-23 | 4 Years | No |
| HH8-W9 | Woman | Junior | 19-23 | 3 Years | No |
| HH9-M8 | Man | Freshman | 19-23 | 3 Years | No |
| HH9-M9 | Man | Junior | 19-23 | 2 Years | Yes |

science, we focused on gender due to the well-documented negative effects of the gender gap in computer science.

We refer to pairs with the label HH-X, where X is the gender of the participant. For example HH3-M5 and HH3-W1 refers to the man (fifth man participant overall) and woman (first woman participant overall) of Pair 3.

## 3.2 Study Design

While the study was conducted in a lab setting, it emulated a remote pair programming environment with the participants seated in separate rooms. Remote pairing was used (1) to create an environment similar to that of our human-agent study and (2) provide benefits comparable to collocated pair programming [11, 49, 67].

Before the study, each participant completed a consent form, background questionnaire, and pre-self-efficacy questionnaire [38]. Instructional video tutorials were given to the participants to teach them concepts of test-driven development (e.g., writing test cases, implementing code, and refactoring code), pair programming concepts (e.g., driver and navigator roles), and think-aloud study (e.g., vocalize any thoughts and feelings as they perform their tasks [103]). The pairs communicated remotely via Teamviewer [163] and used the Eclipse IDE [74] to complete their task.

Prior to the study, the participants were given a warm-up pilot task for password validation. This task allowed for pair jelling — a period of time that allows them to adjust to the pairing. This practice is noted to help pairs work more efficiently in pair programming [98].

All participants implemented a tic-tac-toe game in Java, and completed the task in 40 minutes to avoid any fatigue. In tic-tac-toe, two players take turns placing marks in a 3x3 grid until either one player successfully gets three consecutive marks, or the win condition becomes impossible, resulting in a tie. We used tic-tac-toe due to its relative simplicity - even participants who were unfamiliar with the game could understand the rules. An implementation for a board and the ability to place marks were provided, but participants needed to write test cases and methods in order to complete the game. User stories and acceptance criteria for the task were provided regarding `vertical/horizontal/diagonal wins`, `taking turns`, a `full board`, and a `tie`. Throughout the duration of the task, participants determined their own roles as a driver and navigator. Study sessions were recorded using the Morae screen capture tool [164].

Once the task was finished, participants were asked to complete post-questionnaires on self-efficacy and their pair programming preferences. These questionnaires helped to triangulate (compare and attempt to refute) the results of our qualitative and quantitative analyses.

## 3.3 Data Analysis

Video data, audio data, and screen interactions of participants were collected as transcripts, which were analyzed using a mix of qualitative and quantitative measures, including qualitative methods from Grounded Theory [60] (e.g., Corbin and Strauss variant [159]). Central to qualitative analysis is coding the data – identifying points where certain concepts/phenomena are apparent and marking them accordingly. We coded points based on programmers' utterances to identify key concepts and phenomena via an iterative, open-coding process [90, 145]. This was used to analyze pair behavior. Thematic analysis [20] was used to organize qualitative data into themes that related back to the research questions. For quantitative analyses, we measured productivity, code quality, differences in self-efficacy (before and after the task), and post-questionnaire responses on their pair programming preferences.

The transcribed data from the qualitative study was coded for information such as timestamps, the speaker, their gender, the current driver, type of expression, dialogue relevance to the task, type of knowledge transferred, interruptions, instances of ignoring a partner, collective monologues, and self-disclosure dialogue. The codes were assigned to the utterances of each participant. Three researchers independently coded 20% of the transcripts and reached an agreement on 90% of the coded data by calculating inter-rater reliability using the Jaccard measure [75]. The remaining transcripts were split between the two researchers, who coded them separately.

# 4 HUMAN-AGENT STUDY

To analyze interactions between a human and an agent, we conducted a Wizard of Oz lab study. The study design, as well as the data analysis, was similar to our previous human-human study (Sections 3.2 and 3.3) for conducting comparative analysis.

## 4.1 Wizard of Oz Study

In a Wizard of Oz study, participants interact with a computer agent that is actually controlled by a human "wizard" behind the scenes. Simulating an agent allows for the rapid-prototyping of technically demanding interfaces that have yet to be created, [186] allowing a product's functionality to be iterated before it is later refined based on testing [99, 188]. Wizard of Oz studies are used to develop user-friendly natural-language interfaces, consider the

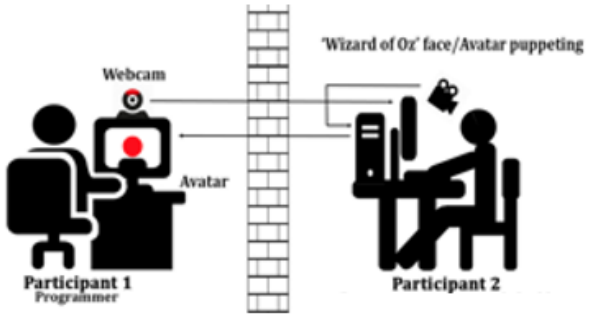


**Figure 1: The design of the Wizard of Oz study. Participant 1 on the left interacts with the wizard (a "PairBuddy") on the right.**

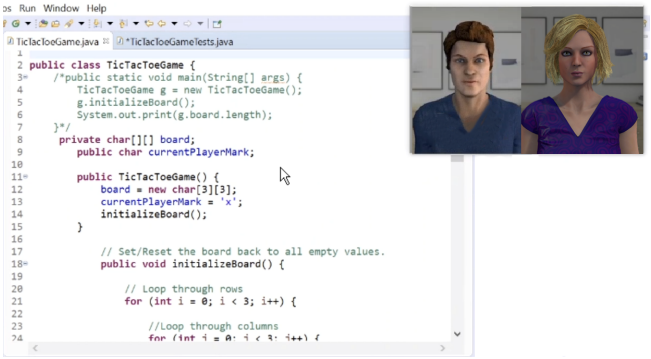


**Figure 2: A screenshot of the agent's interface from the Wizard of Oz study. Participants saw the agent's facial expressions in the form of an avatar and interacted with it via text and voice communication as they programmed in the IDE.**

unique qualities of human-agent interactions [41], and study user interactions with conversational agents [17, 18, 173].

Our study followed a basic Wizard of Oz design as illustrated in Figure 1. Participants sat at a computer to pair program with the agent and used the Saros plugin for the Eclipse IDE to facilitate remote collaboration. Participants interfaced with the agent who was embodied by a 3D avatar (Figure 2) using the Facerig embodiment software [160]. Facerig supported lip-synchronization to the agent's voice generated via Google Text-to-Speech [62]. Each participant's face, voice, and screen were shared with two wizards (Participant 2 in Figure 1). Our agent was collaboratively controlled by a graduate students with 5 years of programming experience and an undergraduate with 3 years experience. They maintained the illusion of the agent by adhering to a templated script of dialogue options. The script helped to avoid any inconsistency in the wizard's behavior when responding to different participants. Skype, Discord, and Google Hangouts were used to facilitate video and audio communication with participants.

The wizards simulated the components of a conversational agent (e.g., speech recognition, intent understanding, dialogue state tracking, dialogue policy, response generation) using a constrained Wizard of Oz protocol [137]. If a participant were to ask, *"How to write code for the win game?"*, the wizard would identify their intent as "implementation help" and choose the appropriate templated response from the wizard's script: *"I can make a recommendation from GitHub, would you like me to do so?"* The wizards controlled both the quantity and quality of our agent's contributions to provide equal benefit to all participants. However, some participants may have received less help, since they denied the agent's requests to drive or ignored suggestions.

## 4.2 Agent Design

In the human-agent study, we simulated a realistic implementation of a pair programming partner agent. This was followed by designing the embodiment (i.e., avatar and voice), as well as a script for the wizard's dialogue based on related literature. Table 2 shows some of the features we implemented for our agent with the relevant sources. The design was inspired by multi-disciplinary research on human-computer interactions, conversational agents, software engineering, human-robotic interactions, education, intelligent tutoring systems, psychology, and management science. Due to the limited available space within this paper, the iterative approach to developing the agent's design decisions are detailed at [138].

*4.2.1 Interactions Using Avatar/Voice/Text/IDE.* Our agent interacted with its programming partners using features such as a dynamic 3D avatar [97, 108, 157, 194], voice [10], and text chat that enhanced human-computer interactions. An avatar makes the interface more human-like [157], thereby improving understanding, engagement, and trust in novice programmers [17, 64, 69, 147, 162, 170, 190]. The programmer could also choose the avatar's gender [19, 35, 87, 165]. For our study, we supported the two most common

**Table 2: Sample Design Guidelines for the Agent.**

| | |
|---|---|
| Having an **avatar** [97, 108, 157, 194] | Clearly **explaining** the reason for yes/no decisions [97] |
| Allowing **gender selection**: man vs. woman avatar [19, 35, 165] | **Expressing uncertainty** with verbal and non-verbal cues [10, 87, 97] |
| Having a **voice** [25] | **Redirecting** suggestions [97] |
| Attributing **success to the group** and taking **responsibility for mistakes** [83, 85] | The **WH** (who/what/where/why/how) question's answers should be accompanied by **directions** and **suggestions** [97] |
| **Adapting** activity based on partner's ability [97] | Making **rapport** by greeting [84] and motivational dialogues [5, 31, 45, 53] |
| Giving **indirect** instructions politely. Giving **direct instruction** as a backseat driver [97] | **Apologizing** when the answer is unknown or a mistake is made [97] |
| Clearly **acknowledging** suggestions [97] | Supporting **verification** after every phase of creativity [97] |
| Showing **agreement** when a human partner is correct [97] | Giving **feedback** related to code quality and productivity with a positive tone [97] |

genders: man and woman. The agent facilitated communication primarily via voice to reduce context switching and increase interactions. Participants communicated with the agent directly through speaking and the agent responded with voice-synthesized messages. Text communication was reserved exclusively for sending links and pictures. The agent directly edited the participant's code using the Saros plugin [154] for Eclipse.

*4.2.2 Making Rapport with Human Partner.* Our agent was designed to build rapport with participants, similar to human-human pairs. It greeted and introduced itself to its human partner at the beginning of each study [84]. Further, it attributed success to the group and took personal responsibility for its mistakes [83, 85]. To increase participants' trust [10, 76, 77, 109], the agent showed uncertainty about its work and asked for verification. For example, after adding the code through the IDE, the agent said *"This might do the trick. I'm not sure though."* Motivation is known to have a substantial effect on a programmer's performance and productivity [5, 31]. Designing extrinsic motivation can help individuals increase their creative and innovative outcomes [9, 45, 53]. The participants were encouraged upon both successes and failures with motivational statements like *"Yay, we did it!"* and *"I'm not sure what we're doing here, but we can always test it."*

*4.2.3 Acting as a Programmer.* Our agent's interactions and capabilities were informed by a wide range of research on automated programming techniques. The agent's ability to contribute code (driver) and give feedback (navigator) were based on automated code/feedback techniques [42, 196] that require a dataset of past solutions and search-based feedback [88, 89, 124, 133]. The agent was designed with the ability to identify unnecessary code - including variables, functions, and classes - based on automated tools like UCDetector [155]. If the participant asked for the location of something within the code, our agent's ability to respond was based on both static [110, 143] and dynamic feature location techniques [106, 143]. For specifically generating test cases, the techniques to generate them automatically (i.e., without past solutions) were either code search algorithms [115, 118] or converting user stories to scenarios, then to test cases [3, 134]. Our agent could also identify where code was missing, a technique investigated by Gerdes et al. [58] in the programming language Haskell.

*4.2.4 Supporting Creativity:* Creativity is vital to a programmer's success [51, 172, 195], especially when solving open-ended problems [21, 43, 102, 107, 130, 178]. To foster diverse thinking, our agent was designed to offer abstract code skeletons/templates (e.g., empty loops and empty function definitions), code examples, and alternative implementations. We then designed questions that encouraged programmers to consciously consider alternative solutions. This design decision was motivated by Tsuei's finding [168] that imperfect guidance about problem-solving strategies and programming concepts enhances creativity and encourages participants to explore new ideas. Since an agent cannot ideate, creativity theory suggests producing a plethora of ideas in order to arrive at creative ones with a concept called ideational fluency [65]. For example, the agent asked, *"Are there other ways we could do this?"*

*4.2.5 Driver/Navigator Communication Styles.* Our agent apologized for incorrect or unknown answers, expressed uncertainty via verbal and non-verbal cues [10, 87, 97], and gave WH (who/what/when/why/where/how) answers accompanied by directions and suggestions. The agent's responses were tuned to its human partner's performance and gave just-in-time help when needed or negotiated interruptions (i.e., when a human partner explicitly asks for feedback in the context of the IDE) [139, 187]. Additionally, our agent provided superior verbal feedback and presented content and code templates to jog the programmer's memory. For example, to correct a programmer, it highlighted code while giving both verbal and visual suggestions; doing so addressed a limitation in human-agent dialogues where self-presentation bias induces a lack of memory retention on the part of humans [91–93, 148].

**Table 3: General Demographics of Participants in Human-Agent Study**

| Participant # | Gender | Level | Age | Experience | |
|---|---|---|---|---|---|
| | | | | Prog. | Pair-Prog. |
| HA-W1 | Woman | Junior | 19-23 | 3 Years | No |
| HA-M2 | Man | Junior | 19-23 | 2 Years | Yes |
| HA-M3 | Man | Sophomore | 19-23 | 2 Years | No |
| HA-W4 | Woman | Junior | 19-23 | 2 Years | No |
| HA-W5 | Woman | Senior | 19-23 | 3 Years | No |
| HA-M6 | Man | Senior | 19-23 | 3 Years | No |
| HA-M7 | Man | Sophomore | 19-23 | 3 Years | No |
| HA-M8 | Man | Freshman | 19-23 | >4 Years | No |
| HA-W9 | Woman | Masters | 30-40 | >4 Years | No |
| HA-M10 | Man | Masters | >41 | >4 Years | No |
| HA-W11 | Woman | Junior | 19-23 | 2 Years | Yes |
| HA-M12 | Man | Masters | 30-40 | >4 Years | Yes |
| HA-W13 | Woman | Masters | 30-40 | 2 Years | No |
| HA-W14 | Woman | Senior | 19-23 | <1 Year | Yes |

## 4.3 Participants

We recruited 14 participants (7 men and 7 women) via snowball sampling, advertisements on social platforms (Facebook and Twitter), and a recruitment site (Upwork). Eight of them were university students (4 men and 4 women) and six were professional programmers (3 men and 3 women). Table 3 gives the detailed demographics of our participants. Since the study was conducted during the COVID-19 pandemic, we conducted virtual lab studies. The choice of a virtual lab allowed us to recruit participants from various time zones throughout the country instead of regional participants only. Similar to the human-human study participants, student participants were incentivized with a $20 Amazon gift card and professionals with a $40 Amazon gift card. The professionals performed the same task, but were given a larger incentive to increase participation.

We refer to the participants in the human-agent study with the label HA-X#, where X is either W or M. For example, HA-W5 refers to human-agent participant 5, a woman, and HA-M2 refers to participant 2, a man.

## 4.4 Study Design

The study design was similar to the human-human study (Section 3.2), except: (1) the participant completed the task with our agent and were given an instructional tutorial on the agent itself, (2) pair

**Table 4: Sample Interview Questions**

| Question |
|---|
| Did you enjoy working with PairBuddy? |
| Do you the trust computer? If given a choice in the future, who will you trust more: human or computer? |
| When PairBuddy wrote code (a.k.a. was the driver) did you trust its correctness? |
| What was your trust level relative to how you would trust a human? Why? |
| Did you trust PairBuddy's advice as a navigator? |
| What was your trust level relative to how you would trust a human? Why? |
| Did you prefer the first avatar or the second avatar? |
| Did you prefer the first voice or the second voice? |
| When working with agents, do you prefer a male or female role? |
| Did you learn anything from PairBuddy? |

jelling was not incorporated, and (3) the agent was set to change gender presentation halfway through each study. We counter-balanced the gender presentation (i.e., whether it was initially presented as man or woman) so that it was equally split among the study sessions.

Further, the study was followed by interview questions to collect the participants' thoughts, feelings, experiences, and preferences regarding our agent. Table 4 gives the sample list of interview questions.

The IRB required we reveal our work as a deception study, so the participants were asked to sign a consent form after the task completion.

### 4.5 Data Analysis

We analyzed the data in the Wizard of Oz study under the same criteria discussed in the human-human study (Section 3.3). Three researchers independently coded 20% of the transcripts and reached an agreement on 93% of the coded data by calculating inter-rater reliability using the Jaccard measure [75]. The remaining transcripts were split among the three researchers. This helped us compare and contrast how a human worked with another human versus an agent; the results were triangulated with interviews and survey data.

## 5 LIMITATIONS

The sample size of 9 pairs for human-human and 14 pairs for human-agent study may be considered small, especially for quantitative analysis. Although the study was gender balanced, we investigated only two genders, men and women, since we did not have any participants of other genders. However, men and women are at opposite ends of the gender spectrum, which provided some insight into gender differences. Although our results showed that our women participants enjoyed an agent more than men participants, our limited sample data set cannot confirm these findings. This needs further investigation to ensure that an agent can benefit men and women equally. Note that there are a variety of factors other than gender that may explain the differences in our results, including participants' personalities, cultural backgrounds, and programming language preferences. The COVID-19 pandemic forced us to conduct virtual lab studies that may differ from collocated studies, but we tried to keep the design as close to the human-human studies as possible – even the human-agent study script was kept close to that of the human-human study.

A different agent design may produce differing results. Though, we considered the main features that a pair programming agent should incorporate, such as negotiated interruptions, a human-like embodiment, divergent thinking, fixing bugs, and adding code. Our results suggested a promising future for pair programming agents.

We only used one simple task, a tic-tac-toe game, to understand the feasibility of a pair programming agent, whereas higher-level programming problems in the industry can be incredibly complex. This research takes the first step towards understanding whether introducing an agent to pair programming is feasible. Our study shows there is potential to increase the effectiveness and efficiency of learning programming concepts with a conversational agent.

## 6 RESULTS

To investigate the trade-offs of substituting an agent in a pair programming context, we compared and contrasted the transcripts, productivity, code quality, and questionnaire results from the human-human and human-agent studies. The results were triangulated with interviews.

In this section, we discuss each research question along with our findings. Each finding is accompanied by relevant key takeaways.

### 6.1 RQ1: Can we continue the benefits of pair programming by replacing a human programmer with an agent?

Pair programming is known to increase code quality, productivity, and self-efficacy, as well as decrease the gender gap. To investigate the similarities and differences between human-human and human-agent interactions, we measured: (1) productivity – the progress made by an individual, (2) code quality – the correctness of the code produced, (3) self-efficacy – an individual's confidence in their ability to perform a particular task, and (4) gender differences for men and women. Finally, we triangulated the findings for RQ1 with the post-questionnaires of participants' pair programming preferences in both studies.

We conducted two analyses. The first was a quantitative analysis to find the significance of the differences between human-human and human-agent interactions using a Wilcoxon signed-rank test. A Wilcoxon signed-rank test is a non-parametric statistical hypothesis test used to compare two samples when the samples' means cannot be assumed to be normally distributed [150]. The second was a qualitative analysis to gain insight into the similarities and differences of human-human and human-agent interaction. As noted, these results were triangulated with interviews.

**(1) Productivity**

Productivity was a measure of how far each pair progressed in the task within the 40 minute time limit in both the human-human and human-agent studies. The task was measured on a 100-point scale; these points were divided equally between test cases and

**Table 5: Human-Human vs. Human-Agent Studies.**

| | Human-Human Studies | | | Human-Agent Studies | |
|---|---|---|---|---|---|
| | **Samples** | **Average** | **P-Value** | **Samples** | **Average** |
| Productivity | 9 | 50.44 | 0.2191 | 14 | 66.00 |
| Code Quality | 9 | 89.33 | 0.9144 | 14 | 98.36 |
| Self-Efficacy | 18 | 3.83 | 0.1093 | 14 | 3.36 |

methods essential for completing the task. The grading metrics can be found at [96].

On analyzing the progress made by 9 pairs in human-human and 14 pairs in human-agent studies, we found an insignificant difference with a p-value of 0.2191 for $\alpha$=0.05 (Table 5). Hence, in both studies productivity rates were similar, although the average score for human-agent (66.00%) was more than human-human (50.44%). In both studies, pairs were more likely to complete the methods (horizontal, vertical, and diagonal) and test cases related to winning. Only two pairs finished both the `tie and take turns` test cases. Further, participants in both studies built off each other's code in order to progress further.

All participants in both human-human and human-agent studies reused code often, as there were methods and test cases that needed few modifications for the tic-tac-toe game. Reusing (copying, pasting, and modifying) code amongst similar sections, such as the horizontal and vertical win methods, helped participants decrease work time.

In the human-human studies, the participants of pair HH3 built off each other's ideas. Similar behavior was found in human-agent studies. For example, the agent added code to a win test case, and HA-W1 then built over its code and added loops.

Though, the dialogues in human-human and human-agent conversations are different; this may have implications for the natural language processing component of future conversational agents. This raises questions about whether machine learning models should be trained on both human-human and human-agent conversations.

**Key Takeaways**

- ***A better sense of direction:*** The agent was particularly useful for keeping participants on the correct path, helping them have a better sense of direction. Conversely, humans may deviate from the path, as was the case with HH6, the pair that scored lowest. Interviews demonstrated how the agent was able to start human participants on a task and keep them on track. For example, HA-M6 said *"So I think having it drive and put in some code helped me like actually start somewhere and get going."* The agent helped participants by starting the task, laying out the groundwork, and giving them direction.
- ***Loss of nonverbal cues:*** Our agent's inability to read and understand human, nonverbal social cues impacted productivity when it acted at inopportune moments. In particular, the agent sometimes interrupted its human partner's thought process regarding the task. For example, in her interview, HA-W9 said *"There are some times when Pairbuddy (our agent) was interrupting me and I forgave Pairbuddy because I knew it was a bot. But I didn't like it."* In human-human collaboration, a human can process a wide range of information, as research has shown that human brains need 200 milliseconds to decipher facial expressions [144] and 0.2 seconds to detect confidence in another human's voice [82]. With current technology, an agent cannot understand their partner's facial cues and vocal intonation on the same level as a human, nor can it communicate in these ways itself. Therefore, nonverbal interruption is lost as a communication method. Humans must take time to communicate their thoughts and ideas verbally in order for the agent to understand them.
- ***Lack of extended discussions:*** The length of the discussions between the partners as driver and navigator were transformed with the introduction of an agent. In human-human interactions, larger-scope errors were discussed in more detail but took longer to resolve as the participants talked through the issue. In HH3, the participants discussed an error where the game board would not print. They discussed the potential solutions for this error at length (3.01 minutes). Although these discussions are important, they hindered productivity in human-human studies.
  Comparatively, in human-agent interactions, the agent could point out errors in the code and give their locations; however, our agent would often fix them without full explanation. For example, the agent added the code template for a `win` test case without explanation as to what code was added and why. A carefully designed agent may be able to provide explanations for the decisions.

**(2) Code Quality**

To evaluate whether the code produced from the human-agent study was of the same quality as the human-human study, we measured the code quality on a scale of 0 to 100. The metrics can be found at [96]. We measured productivity and code quality separately, then combined them for a final grade. For example, in the human-human study, HH5 had a productivity score of 53% and a code quality score of 87%. Therefore, we calculated 87% of 53% for a final score of 46.11% (53 * 0.87 = 46.11%).

On comparing nine human-human and fourteen human-agent studies using a Wilcoxon signed-rank test, we found insignificant differences with a p-value of 0.9144 for $\alpha$=0.05 (Table 5). On average, human-agent studies' code quality scores were 12.03% higher than human-human scores (98.36% human-agent vs. 89.33% human-human).

Both humans and the agent helped their partners improve the usability and readability of their programs by refactoring, fixing simple issues, and organizing/structuring code. For example, human-agent partners would catch typos, repetition of variable names, and missing brackets or semicolons. Human-agent partners also helped improve organization and readability by properly formatting new

and existing code. In one case, the agent made a suggestion to HA-W5 by saying, *"I think we have a syntax error. I suggest we format the code,"* and on HA-W5's consent, proceeded to format the code for her. This type of correction occurred in 5 human-agent studies and 3 human-human studies. Similarly, both types of partners helped to assist with accidental repetition of code for the methods or the tests.

**Key Takeaways**

- ***Quick access to a plethora of online sources:*** The programming agent's ability to utilize online sources, such as Github and Stack Overflow, or past code examples can be utilized to improve the code quality. Our studies also found that our agent's familiarity with the task made the agent more effective at performing the task at hand. Thus, it was able to give more useful hints and advice to its partner, while a human partner was more likely to propose and follow an inefficient solution. In HH6, the participants pursued an ineffective solution throughout the study, leading to a code quality score of 18%. They did not recognize their mistake, making comments like *"that looks good"* throughout the duration of the task.
  We designed our agent to help code more accurately, provide motivation, and direct to potential errors. Whenever the agent saw that participants were stuck with the code, it would offer to help. For example, the agent told HA-W11 that *"If you need any help getting started, I can try,"* to which she responded *"Sure, I can actually use some help getting started here."* Since the agent had more information on the task, humans treated it accordingly. Sometimes, the agent would also encourage its partner to ask more questions with statements like, *"I feel unsure about the approach we took here."*
- ***Different code contribution style:*** Our agent provided the code as skeletons and templates, while, in human-human studies, participants added completed sections. Instead of providing logic verbally, our agent gave some indication of logic via the structure of the code. While driving, our agent provided code skeletons for humans to interpret and fill out. While our agent's logical limitations sometimes made it difficult for the programmers to put its suggestions into effect, it was able to guide its partner via code additions. In interviews, when asked how our agent helped, HA-M10 said that *"it puts at least the skeleton there, of the Boolean logics and everything."* This design decision is specific to our agent; it was chosen to help programmers get started and think about diverse solutions for the problem in hand.

**(3) Self Efficacy**

Self-efficacy refers to a person's belief in their own success. One major benefit of pair programming is an increase in morale among students [179]. To investigate whether pair programming helped increase the self-efficacy of the individuals, we measured the differences between self-efficacy scores before and after the task in both the human-human and human-agent studies.

On comparing the self-efficacy scores between 18 individuals in human-human and 14 individuals in human-agent studies, we found insignificant results with a p-value of 0.8339 for $\alpha$=0.05. In both studies, the average self-efficacy scores of the post-questionnaire were greater than those in the pre-questionnaire (see Figure 3), indicating that having either a human or agent partner helps in increasing self-efficacy.

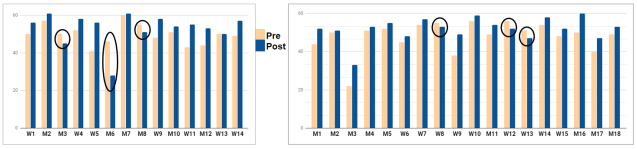


**Figure 3: Human-Agent and Human-Human Studies Participants Self Efficacy. The circled participants had decreased self-efficacy.**

Verification from a partner may increase self-efficacy. In both studies, one member of the pair verified their partner's code.

**Key Takeaways**

- ***Validation of tasks:*** Confirmation from the agent was particularly reaffirming for some participants as they gained validation from the agent. While humans could validate each other, they may not have a full understanding of the task. HA-W1 said, *"because it was automated for the program, it kind of helped confirm that I was doing the right thing."* Validation from an agent may be perceived as more valuable if it is automated for a relevant purpose [34, 72].
- ***Loss of complex discussion:*** The complex discussions and interactions that can happen with a human partner were lost with an agent. As HA-M6, the participant with lowest post-self-efficacy, said, *"There wasn't as much human interaction, and it was hard to feel like I was actually knowledge sharing or trying to understand what we were tackling."* One of the other participants who ranked lower in post self-efficacy, HA-M3, said in his interview that *"not being able to complete a single one of those stories is a hit to my self confidence."* Human-like discussions with an agent are not possible with current technology.
- ***Interpretation of code:*** The participants had to depend on their own interpretations, since our agent did not explain the code. As HA-M3 said, *"It (the agent) wrote the code well, and the code that it wrote was functional for what it did, but it never actually explained what it drove and left it up to my interpretation."* Conversely, human participants were able to explain their code if prompted.

**(4) Gender Effect on Pair Programming Measures**

We investigated the differences between our men and women participants while working with both human and agent partners. When comparing men in human-human (9) and human-agent studies (7) we found no significant change for productivity, code quality, and self-efficacy (see Table 6). Our women participants had a significant difference in productivity (p-value= 0.01055 for $\alpha$=0.05), and a partially significant difference in self-efficacy (p-value= 0.0626 for $\alpha$=0.05). On comparing averages, both men and women participants had positive experiences with our agent, but women benefited more than men (see Table 6). Usability/reusability and readability of code, as well as verification, helped both genders, as seen in Table 6.

**Table 6: Men vs. Women**

| | Men | | | Women | | |
|---|---|---|---|---|---|---|
| | **P-Value** | **Average (HH)** | **Average (HA)** | **P-Value** | **Average (HH)** | **Average (HA)** |
| Productivity | 0.832 | 54 | 55.57 | 0.01055 | 42 | 76.43 |
| Code Quality | 0.7141 | 99.67 | 98.57 | 0.9031 | 95.67 | 98.14 |
| Self-Efficacy | 0.08892 | 5.33 | -1.4286 | 0.0626 | 2 | 8.1428 |

**Key Takeaways**

- ***Women participants' self-efficacy increased:*** Throughout our studies, women scored significantly higher in post-self-efficacy when working with our agent rather than a human; averages were also higher for the women participants working with the agent when compared to men (Table 6). This is an interesting shift, as men consistently score higher than women in self-efficacy ratings when working in areas perceived as "masculine," such as STEM [128]. Further, in previous human-human pair programming research, men's average post-self-efficacy was greater than women's, regardless of their partner's gender [87].
- ***Different preferences for agent's gender:*** As demonstrated in previous research, women in human-human pairs usually prefer to work with other women, while men did not have a gender preference [87]. There were mixed opinions from participants of both genders on the avatar and voice of an agent partner, as seen in Figure 4. A study on gender presentation for conversational agents revealed that they were more likely to be characterized as female due to projections of existing gender stereotypes [39]. As women are more often associated with personal assistant roles, bots with similar assistant-like roles such as Siri and Alexa are presented and interpreted as feminine [39]. However, with a pair programming agent, we conjecture that the male-dominated state of the computer science field may further effect gender preferences.

*6.1.1 Pair Programming Preferences.* Our results demonstrated the feasibility of replacing a human partner with an agent. We conducted quantitative and qualitative analysis in order to measure changes in productivity, code quality, and self-efficacy. These were further triangulated using the post-questionnaire pair programming preferences.

The post-questionnaire on pair programming aspects for both human-human and human-agent studies helped us measure individual's preferences for working with a human or our agent. On conducting a Wilcoxon-signed rank test with the pair programming preference scores of 18 individuals in human-human studies and 14 individuals in human-agent studies, we found insignificant p-values for most of the questions. The only significant p-value of 0.00652 for $\alpha$=0.05 was for question 12, which was "pair programming with test-driven development works for me." This significant value suggests that our agent helped participants learn the concepts of test-driven development and adjust to pair programming. Figure 5 shows the average scores (y-axis) in human-human studies (red line) and human-agent studies (black line) for each pair programming preference question on the x-axis.

The trend line in Figure 5 shows similar sentiments from human-human and human-agent studies about various aspects of pair programming. This shows that our agent helped participants with getting started and understanding the format of the task.

Reflecting our results in RQ1, participants also scored themselves higher on code quality and productivity. On average, human-agent participants scored themselves higher than human-human participants for higher quality code, better testing and debugging, confidence in solution, and enjoyment of technique (see Figure 5). They also scored lower for inefficient use of time and interruptions distracting both partners (Figure 5).

On comparing questionnaire results by gender, we saw an interesting shift between human-human and human-agent studies. As seen in Figure 5, women participants had better scores for every pair programming question while working with our agent in comparison to a human partner; their scores increased by over a full point for "enjoyment of technique" and "confidence in solution." Conversely, men participants' scores either stayed the same or decreased when our agent was introduced.

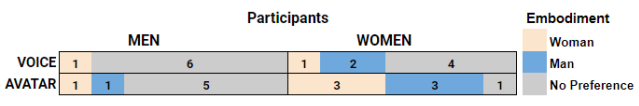


**Figure 4: Participant preferences on voice and avatar.**

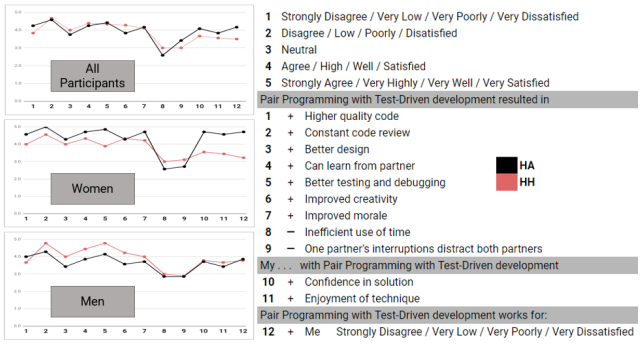


**Figure 5: Pair Programming Preferences Charts. The red line represents human-human averages, and the black line represents human-agent averages. The questions along the x-axis correspond with the numbers in the legend to the right. The answers were given on a scale of 1 - 5 ; 1 being the most negative response at Strongly Disagree / Very Low / Very Poorly / Very Dissatisfied and 5 being the most positive response at Strongly Agree / Very High / Very Well / Very Satisfied. The (+) sign signifies that a higher score is preferred, while a (-) sign signifies the opposite.**

## 6.2 RQ2: What kind of knowledge is transferred between human-human and human-agent pairs?

One important benefit of pair programming is knowledge transfer, or individuals exchanging relevant information about the task with each other. As illustrated by Cockburn et al., *"Knowledge is constantly being passed between partners, from tool usage tips (even the mouse), to programming language rules, design and programming idioms, and overall design skill" [37].* The code set used for knowledge transfer is detailed in Table 7. The Tool, Program, Bug, Code, and Domain code sets for knowledge transfer were inspired by Jones and Fleming's study [98]. In our studies, we found an additional knowledge transfer code set, Technique. Table 8 gives the frequency of these knowledge transfers in our human-human and human-agent studies, as well as knowledge our agent provided in human-agent studies.

As the given task was centered around the use of test-driven development, it was important that participants understood the technique they were required to use. Our agent was programmed to give suggestions, such as telling the participant to write test cases for test-driven development. Many participants, particularly among the university students with less overall experience, were not as familiar with test-driven development in Java. Therefore, this was an important knowledge transfer to make. These results were also supported by the post-questionnaire on pair programming preferences (discussed in RQ1). Another technique was creating test cases based on the user stories, and, in our studies, both humans and the agent were noted to direct their partner towards making tests, as per the user stories.

Further, both humans and the agent directed their partners to the location of bugs/errors in the code that may otherwise have gone unnoticed by the other participant. Our agent noticed 42 of the 98 bugs in human-agent studies; it either informed the partner or directly fixed the bug/error. For example, HA-M7 asked *"Can you point me in the right direction here?"*. The agent responded *"Between line 74 and 75, add an if statement."*

Participants transferred knowledge regarding programming concepts and organized their own knowledge as a result of interactions in pair programming [27, 28, 104, 105, 113, 129, 131].

In pair programming, pairs learned topics better while working together [181], narrowed their gaps in knowledge [8, 36], exchanged both project-related and general knowledge [132], and discovered new tools [121]. We found similar knowledge transfer in human-agent studies. For example, when asked what he learned from our agent, HA-M2 responded, *"yeah, about formatting, how to do test-driven development a little bit. I mentioned wanting to do the test-driven part of it, but he or it or the robot helped me make sure that I was writing it correctly. And I liked that."*

**Key Takeaways**

- ***Getting unstuck:*** Human-agent pairs gained the ability to get unstuck with comparable ease, as our agent provided skeleton or base code. The agent would provide empty methods for the task, properly formatted empty loop shells and example test cases. Code skeleton generation is shown to improve consistency and reduce errors [16]. For example, HA-W11 said, *"I will say having this example right here already there helps a lot."* Conversely, in human-human studies, a detour in logic sometimes derailed the entire task. The knowledge transfer from our agent also expedited the completion rate (see RQ1).
  The agent was designed to make helpful suggestions to the participant in order to to spark creativity on their end, most commonly, *"Are there other ways we could do this?"* Rather than presenting a potential solution in full like a human partner, our agent would facilitate learning and problem solving by helping the human formulate an answer. Encouraging a variety of ideas and potential solutions helps to facilitate learning [80, 81].
- ***Fostering creativity:*** Our agent was designed to pursue a specific solution to the problem, while human participants pursued varying solutions. Although participants in the human-agent studies had the choice of pursuing an alternative idea, they were less risk averse and continued with the agent's solution. Various factors such as education level, gender, and agent behavior further impact trust, and some users may over-rely or under-rely on the agent [34, 72]. HA-W1 said, *"I trusted the computer because I assumed it knew what the right answer was."* Research has shown that integrating a virtual agent into the design of artificial intelligence programs increases user trust [174]. Although, as discussed in RQ1, this was effective, as our agent usually helped them solve the problem correctly, it can nevertheless be perceived as dampening creativity.
- ***Lack of logical explanations:*** While the participants in the human-human study frequently explained the reasoning and logic behind their decisions to their partner, the agent was not capable of this. If asked to explain how its code worked, it would simply reply, *"Sorry, I'm not good at understanding logic."* In human-human studies, the participants were observed building off each other's knowledge and logic via discussion. Research has shown that collaborative learning is more effective than traditional methods such as lectures, as it allows students to build their own mental models based on discussion and knowledge transfer that occurs during the problem-solving process [2].
  Typically, human navigators will contribute ideas to the task while their partner is coding [98], but our agent would only ask the participant what they were doing and correct errors, making it somewhat inefficient as a navigator. For example, HA-M6 said, *"We never really discussed the solution... (why we were) doing something... or what might be helpful."* Unfortunately, our agent could not provide this type of detailed discussion [177]. Generating effective explanations and discussions is an active research area (e.g., [14, 117]), and this may remain one of the major challenges of replacing a human with an agent.
- ***Unidirectional knowledge transfer:*** Knowledge transfer from a human to the agent was not possible. While human-human pairs learned and experienced knowledge transfer simultaneously as they worked, an agent's knowledge cannot and did not evolve during the task.

**Table 7: Knowledge Transfer Definitions and Examples**

| Knowledge Transfer | Definitions | Examples (from our study) |
|---|---|---|
| Tool | Knowledge about the IDE or how to use the tool. | *"there's a key bind for going back a tab"* |
| Program | Knowledge about the programming language itself or it's syntax. | *"I could also just aggregate the chars. I think they work like strings in that way."* |
| Bug | An error in the code. | *"I think we are missing code in this function."* |
| Code | The code itself; ie. what they are programming. | *"So I'm gonna need to use a for loop as well."* |
| Domain | The task (in this case, tic-tac-toe game) | *"The goal is to get three of...marker...in a row."* |
| Technique | About the techniques being used (i.e., test driven development, pair programming) | *"So if it's test driven development, I want to start by writing a test, I think."* |

**Table 8: Knowledge Transfer Frequency in Human-Human (HH), Human-Agent (HA), and from the Agent.**

| Knowledge Transfer | Frequency in HH | Frequency in HA | From Agent |
|---|---|---|---|
| Tool | 5 | 0 | 0 |
| Program | 15 | 2 | 0 |
| Bug | 48 | 98 | 42 |
| Code | 672 | 437 | 56 |
| Domain | 14 | 6 | 0 |
| Technique | 6 | 15 | 6 |

## 6.3 RQ3: Do human programmers consider the agent as their partner?

We investigated human attitudes towards the agent in comparison to other humans when:

**(1) Interrupting the Partner**

An interruption is "a starting up of some intervention by one person while another's turn is in progress." Interruptions are associated with interpersonal dominance [1, 120, 192], friendliness [123], engagement [59], and involvement in the interaction [151].

In pair programming, one important benefit is getting help from a partner. Both human and agent partners were able to provide help, and the way in which participants asked for help was consistent between human-human and human-agent interactions. Both used interruptions to either ask for or offer help. In general, human programmers tended to ask for help indirectly rather than directly stating that they needed help. For example, in HH7, HH7-W8 said, *"but I'm not sure how to do that."* This behavior stayed consistent when humans were working with our agent. For example HA-M8 said, *"I don't know what to do from here."*

**Key Takeaways**

- ***Willingness to ask for help and address uncertainties:*** Participants were less self-conscious about mistakes and uncertainties while interacting with the agent; in contrast, humans are hesitant to ask for help from a human partner, as they may feel week, needy, or incompetent [169]. As HA-W5 said, *"I think the big thing is sometimes as a programmer, it's embarrassing when you make mistakes. So you're stuck on something around your colleagues, but PairBuddy (our agent) I don't think judges me."* This may have influenced the productivity and code quality, as seen in RQ1.
- ***Lack of diversity in responses:*** However, due to the limits of current technology, a wide range of responses were lost. Once the human partner got their first response addressing our agent's limitations regarding logic they asked simpler questions (e.g., code implementation or test related). For example, HA-W14 asked the agent a complex question regarding logic, and the agent responded, *"I'm not good at logic."* From then on, the participant only asked non-logic related questions about the next steps or switching roles. The limitations of an agent can discourage asking questions when the participant doubts the agent's capability to answer, though, different agents may be designed with varying capabilities and responses.
- ***Providing negotiated interruptions:*** This feature was dependent on the design of our agent. If an agent is designed differently, then this behavior may change. In human-human interactions, the helper was more likely to either help without asking first, or their partner would ask for them to help code. The agent never did anything to the code without asking first, so it could only give help when explicitly permitted to do so. It would prompt the change in roles with questions like *"mind if I drive?"* or *"would you like some help?"*. Therefore, our agent could only help with the code as much as the participant allowed. Our agent was intentionally designed to have negotiated interruptions [78, 139, 187]. This could hinder productivity by distracting a human mid-thought, but it could also assist productivity, as negotiated interruptions are designed to give useful advice.

**(2) Building on Partners' Ideas**

In collocated pair programming, individuals act upon, dismiss, or modify/refine their partners' ideas [98]. As remote pair programming shows similar results to collocated, we found this behavior in our human-human study.

In human-human studies, we primarily saw participants act on and modify ideas, with less dismissal. For example, participants in

HH2 acted on and modified each other's ideas as they worked. Similarly, in human-agent studies participants acted on and modified the code skeletons provided by our agent, but did not refine them. For example, Figure 6 shows how HA-M2 changed the agent's given code skeleton.


**Figure 6: Instance of HA-M2 adding onto the code skeleton provided by the agent.**

**Key Takeaways**

- ***Ignore agent without a social cost:*** There were 22 instances of human participants ignoring the agent, but only one instance of a human ignoring another human. HA-M8 said simply, *"I did not consider (the agent's) feelings."* In contrast, a human partner may have to bear the social cost of ignoring or disagreeing with another human's suggestions. HH3-M5 said that *"I just want them to, like, be semi-polite about it (corrections)."* Where ignoring a human may lead to discomfort and hurt, an agent will not have any emotional conflict with its partner.
- ***No acknowledgement to agent's uncertainty:*** While coding, the agent made statements like, *"This might work. I'm not sure, though."* However, human programmers did not acknowledge the agent's uncertainty. Since agents can still make errors, human programmers need to stay vigilant while working with an agent. They also ignored when the agent expressed uncertainty about how the human's code worked.
- ***Not refining agent provided code:*** Further, in human-agent studies, participants accepted the agent's ideas. Though, since they did not modify/refine them further, their solutions may have lacked diversity. In human-human studies, participants would modify and refine their partner's code when necessary.

**(3) Trusting a Human Partner vs. an Agent**

One important factor of pair programming is trust; partners need to consider and accept suggestions and input from each other. Our participants built trust with both human and agent partners over time. HA-W5 said, *"I think I was a little distrustful at first of the application, but then after working with it, especially after I saw that it was helping me solve the problem, I trusted it a little more."* Similarly, in the human-human studies, HH3-M5 said, *"at the very beginning I didn't (trust) just cause it was a stranger, but as it (time) went on... she's doing what I was planning and then... I didn't have to steer so much."* Hence, humans gradually built trust with both human and agent partners.

**Key Takeaways**

- ***Trusted agent with simple programming tasks but not complex tasks:*** Participants tended to take the agent's advice most of the time for our task. For example, HA-M8 explained why he felt inclined to trust the agent, saying, *"the computer knows more simple functions than a human knows. In that case I would trust a computer more."* This demonstrates that trust was easily built with an agent for simple programming tasks.
  With the current state of artificial intelligence, the capability to handle complex tasks was lost. Four participants expressed that they would prefer to work with a human for more complex tasks. For example, HA-M6 said that he had a slight preference towards human partners because *"it (the agent) doesn't give that more specific, explanatory approach."* Nevertheless, they also expressed that our agent would be more useful for simple tasks.
- ***Trust based on agent's embodiment:*** Participants' trust levels for the agent were transformed based on the agent's embodiment. Research has shown that human-like agents are more easily trusted [146]; therefore, the voice and avatar used to anthropomorphize our agent may have influenced the participants' trust levels. Four participants noted that they would enjoy working with our agent more if it was more human-like in demeanor. For example, HA-W4 expressed that the dialogue should be *"more human - then you feel like you're sitting there with a person."* However, three participants expressed that being too close to a human would seem uncanny. As explained by participant HA-M6, *"I think there's something a little creepy about something that's really close to being human...I'm more comfortable with it being a little more obviously a robot."* These results are similar to literature wherein empirical evidence is mixed about the necessity of the embodiment of an agent [46, 69, 70, 119, 170, 190].

**(4) Humility Towards a Partner**

We measured the humility towards partners based on the usage of collective monologue and self-disclosure dialogue. Humans use collective monologues when working in social environments, using plural pronouns like "us" and "we" more frequently than when alone. In both pair programming studies, pronoun choice was analyzed to see if humans considered the agent as a partner. Participants had the same frequency for using singular "I" or "you" pronouns between the human-human (avg. 37.3%) and human-agent studies (avg. 35.2% ) and, in turn, they had similar ratios of plural "us" or "we" pronouns (64.8% for human-human and 62.7% for human-agent). Further, in both human-human and human-agent studies the participants used the same grammar in similar situations. For example, when their partner would make a mistake, participants would switch to a singular pronoun to acknowledge the mistake. HA-M2 said to the agent, *"I can't tell if your code…"* Similarly, HA-W9 said to the agent, *"You don't have the test…"* Humans also acknowledged their own mistakes with singular pronouns while working with the agent. For example, HA-M3 said, *"I made a mistake on line 33..."* Hence, participants showed the same humility towards our agent as towards a human partner.

**Key Takeaways**

- ***Adding human-like humility:*** If desired, a similar level of human-like humility can be incorporated into an agent's

**Table 9: Summary of Our Findings. (*) Shows Phenomenon Specific to our Agent Choice.**

| # | | Similar Aspects | Gained | Lost | Transformed |
|---|---|---|---|---|---|
| RQ1 | **Productivity** | Reuse code | Correct path | Non-verbal cues | Length of discussions |
| | **Code Quality** | Usability; readability | Online resources | N.A. | Code contribution style* |
| | **Self Efficacy** | Verification | Better validation | No complex discussions | Interpretation of code* |
| | **Gender** | Reuse code, usability, readability, verification | Helped women | N.A. | Avatar/voice preference |
| RQ2 | **Knowledge Transfer** | Bug location; test-driven development; organize code | Unstuck; creativity | Explaining logic | Unidirectional transfers |
| RQ3 | **Interrupting** | Asking for help | Less self-conscious | Agent's limited responses* | Interruption style* |
| | **Building on Ideas** | Acting on and modifying ideas | Ignore without social cost | Ignore agent's uncertainty | Refining partner's ideas |
| | **Trust** | Built over time | Trust for simple tasks | No complex task solved | Based on embodiment |
| | **Humility** | Used plural pronouns to show partnership | Dialogue design can bring humility* | N.A. | Addressed as person and thing |

design. This can be achieved by integrating dialogue templates that consider collective monologue for collaboration and self disclosure for admitting its mistakes.

- ***Acknowledging success as a team and self-disclosure for mistakes:*** We wrote our script to encourage human participants to see their relationship with our agent as a team. Therefore, as stated, we ensured that our agent contributed successes towards the team, and its mistakes towards itself. This worked well, with the "I" vs. "We" ratios staying consistent between human-human and human-agent studies.
- ***Addressing agent as a person or a thing:*** Third person pronouns such as "he," and "she," were used for our agent, but participants also called the agent "it." For example, HA-W11 referred to the agent as "it" when she said *"Awesome. So it (agent) just did that same change again"*. She also discussed the agent in third person while working with it: *"As I was saying before, I think I'm learning more how to talk to this bot (agent)."* It was clear that the humans in the study recognized the non-sentience of the agent partner and addressed it accordingly.

# 7 TRADE OFFS OF AN AGENT

Our results evidenced the feasibility of using an agent as a programming partner. Table 9 summarizes our findings regarding: (1) What aspects of pair programming were similar between human-human and human-agent interactions, (2) What new aspects of pair programming were gained with an agent? (3) What aspects of pair programming were lost with an agent? (4) What aspects of pair programming were transformed by the introduction of an agent? We conjecture that these results can be generalized by designing an agent informed by research from disciplines such as intelligent tutoring systems, human-robotic interactions, psychology, education, and software engineering. The primary factors that may have affected our results were the way the agent contributed code as a driver, how it provided negotiated interruptions, human interpretation of code in lieu of explanations from the agent, agents' limited responses regarding complex logic questions, and the dialogue script for adding humility. Hence, as suggested by our results, a pair programming conversational agent brings its own set of good, bad, and ugly aspects.

## 7.1 The Good

**(1) A Non-judgemental Partner Addressing the Pipeline Problem.**

A conversational agent, utilizing artificial intelligence and machine learning technologies, can resolve the problems associated with human pair programming partners, such as scheduling, collocating, imbalanced roles, and power dynamics. Two participants, HA-M2 and HA-W5, noted that they felt more comfortable with the agent, since it would not judge them (see Section 6.3). This shows that it can act as a non-judgemental partner, which will be especially beneficial for programmers who may be reluctant to form partnerships with their peers, such as people who are introverted, autistic, or minorities like women and people of color. Since computer science is not a diverse field, it can be difficult for these people to find pair programming partners they feel comfortable with. Therefore, a non-judgemental pair programming agent could help these groups feel welcome in computer science and encourage diversity.

A non-judgemental agent has the potential to encourage diversity; our results showed a significant increase in productivity and self-efficacy scores among women participants when working with an agent. The difference in these scores could be influenced by mixed gender pairs in the human-human studies, as Kuttal el al. [87] found that, in human-human pair programming, women showed lower self-efficacy scores when working with men. However, comparisons of pre- and post-self-efficacy questionnaires reveal that some men participants showed a decrease in self-efficacy with an agent. This discrepancy may be attributed to other factors; nonetheless, the agent's design should promote gender equality by utilizing methods like GenderMag [23, 24, 40, 71, 171], which removes gender biases from agent designs. GenderMag helps find and

fix gender-related issues in problem-solving software and increase gender inclusiveness.

Such a conversational agent will need to be trained on data from various populations to integrate diversity. Machine learning algorithms can introduce bias in their results if they are not trained with diverse data [101]. With the current under-representation of women and minority groups in the computer science field, it may be difficult to find diverse populations of programmers [158]. If we are not careful with data selection, it may further perpetuate stereotypes about the field.

Further, an agent – being a low-cost interactive programming partner – can help solve the pipeline problem and facilitate programming education effectively, efficiently, and comfortably for students. Such an agent can easily be integrated into interactive education platforms such as Codecademy [153] and intelligent tutoring systems (e.g., [2, 32]).

Such integration can transform education for people in rural areas and students learning remotely, including during a global pandemic like COVID-19. With the various pitfalls of online communication including poor internet connection, lack of resources, and scheduling conflicts, it is good to have an alternative option for collaborative work scenarios.

**(2) Motivating Partner**

For human-human pairs, the amount of motivation participants would give their partner depended on their individual personalities. Our simple design using motivational feedback was effective when the human-agent participants were struggling, making progress, or about to give up. Motivation and assistance are important aspects of student learning, as supported and researched in intelligent tutoring systems [12, 44, 48, 111]. The agent's motivation and feedback helped students achieve their goals and scaffold self-regulated learning [48]. Rebolledo-Mendez et al. [135, 136] created motivational models based on effort, independence, confidence, and level of attention that were later implemented by existing intelligent tutoring systems; these models are able to provide consequent scaffolding. Utilizing the literature about intelligent tutoring systems, we see a great potential for achieving increased motivation among programmers in a pair programming context.

**(3) Active Learning Support**

We designed the agent to support active learning instead of a tutor-tutee dynamic, similar to computer-supported collaborative learning tools. Although these learning tools provide an exceptional learning environment, they require at least two participants and do not offer individualized assistance and guidance [61]. Agents can be designed to have the benefits of intelligent tutoring systems [4], while simultaneously providing pair programming benefits.

The "free rider effect" may emerge in pair programming when one individual in the pair performs poorly, partners do not exert maximum or equal effort, or the pair fails to coordinate optimally [50, 57, 193]. To alleviate the free rider effect, frequent role exchange is recommended [57]. Agents can be designed to promote equal pair programming by measuring statuses like time, effort, and progress. Therefore, our agent was designed to balance roles by tracking the time spent driving for each participant, and when the participant showed free rider behavior, the agent would politely decline to drive. For example, HA-W14 repeatedly asked the agent to drive, so the wizard eventually declined the offer.

**(4) Supporting Creativity and Problem Solving**

The agent's design can be enriched to include creative problem solving processes and problem-solving strategies. Examples of these such as clarifying the task, generating ideas, developing solutions, and implementing the solutions were made to the extent possible with current technology (detailed in Section 4.2). Further, agents can utilize problem-solving strategies such as divide and conquer, analogy, generalization, backwards, and "sleep on it" [102, 130, 178] by using textual hints or by providing code examples, such as an Idea Garden [80, 81]. Agents could be designed to accommodate different programming and learning styles based on gender as well. For example, women are more likely to see navigator as the primary communicator role, while men interpret the driver as the primary communicator [87].

## 7.2 The Bad

**(1) Excessive Trust Towards Agents**

Humans trusted the agent to a notable extent, which caused them to become risk averse, as users tended to make decisions that maximized efficiency at the cost of thoroughness [54]. The higher trust by users in the agent can lead to automation bias and greater situational trust in an automation [73, 161]. We saw similar behavior for programmers while pair programming with our agent. For example, HA-W9 said, *"I trusted the code that (the agent) gave me because I knew it (the agent) already knew the answer."* Such behaviour led to inflexible solutions produced in the programming effort and could effect learning outcome, especially for students.

**(2) Lack of Explanations**

Our agent was unable to explain the logic behind the solutions or code it added to the program. HA-M6 stated his preference for working with a human, as he wanted the agent to have a *"more specific, explanatory approach like a human."* Similarly, Simon and Susan Snowden [152] found that students are unable to understand the purpose of code or recognize it without explanation. While there is a potential to utilize tools like Whyline [94, 95] to answer debugging related questions by visualizing the runtime events, generating feedback and human-like explanations is not possible. Artificial intelligence research already aims to produce more explainable models to enable users to understand how an agent arrived at a solution [14, 117]. Our research illustrates the importance of generating explanations for conversational programming agents.

## 7.3 The Ugly

**(1) Lack of Interpersonal and Social Skill Support**

An agent cannot completely replace a human in a pair programming context. Some of the interpersonal and social characteristics learned with another human cannot be learned from an agent. Pair programming with another human is helpful for getting a glimpse into real-world collaboration, communication, and coordination, as well as broadening knowledge, gaining a greater sense of responsibility, forming friendships, and making other helpful connections [30, 52, 87, 112, 112, 131]. These benefits may not be completely replicated with an agent.

In human-human collaboration, the human brain can quickly process large ranges of information based on facial expressions and vocal intonation [82, 144]. With current technology, an agent

cannot understand or replicate these forms of communication in an efficient manner, nor can it tell which interruptions are received positively and which are received negatively. As previously mentioned, many factors influence whether an interruption is perceived as cooperative or disruptive [59]. However, an agent cannot distinguish between these indicators, nor can it understand how they affect their partner's perception.

**(2) Lack of Educative Conversations on Program Logic**

Humans discuss with each other to resolve issues during pair programming. However, such in-depth discussions on programming logic are not currently viable with conversational agents.

## 8 CONCLUSION

This research lays the groundwork for the feasibility of pair programming agents. Based on our quantitative and qualitative analysis of the human-human and human-agent studies, we found:

- *RQ1: Can we continue the benefits of pair programming by replacing a human programmer with an agent?*
  On comparing the scores for productivity, code quality, self-efficacy, and pair programming preferences, we found no significant differences. Participants enjoyed learning test-driven development concepts using an agent. The results also showed that women in particular benefited from the introduction of an agent.
- *RQ2: What kind of knowledge is transferred between human-human and human-agent pairs?*
  The agent was unable to explain its logic, or elaborate on their partners' ideas, whereas human partners could analyze the logic behind agent-provided code themselves to extract the ideas. However, the agent was able to provide code skeletons and hints to lead their partner towards a solution.
- *RQ3: Do human programmers consider the agent as their partner?*
  Human partners trusted the agent instructions often without any questions, in one case responding, *"I'm going to blindly believe you."* Human partners interrupted agents when they did not know what to do next, were stuck, were unsure about their problems, or wanted clarification, whereas human-human pairs were more hesitant and asked follow-up questions to understand, clarify, or verify their partners' decisions. Also, human partners showed the same humility towards agents and humans by attributing success to the group using words like "we," and taking personal responsibility for mistakes using "I" or "me." Human partners acted on or dismissed the agent's ideas as humans do, except that humans tended to modify/refine their human partners' ideas.

A pair programming conversational agent led to *the good* – an agent can act as a motivating, non-judgemental partner that can help solve the pipeline problem by allowing active learning–*the bad*– humans tend to blindly trust agents, and current technology necessitates new ways of generating diverse solutions to the programming task in-hand along with explainable feedback –and *the ugly*– a lack of social and interpersonal skills that can be only learned with another human and the inability to hold complex discussions on logic and ideas. The positive results proved the feasibility of developing an Alexa-like programming partner that could change the future of programming and computer science education.

## ACKNOWLEDGMENTS

We would like to thank the CHI reviewers, especially Luigi De Russis, for his insighful feedback. Thanks to our study participants as well.

## REFERENCES

[1] Piotr D. Adamczyk and Brian P. Bailey. 2004. If Not Now, When? The Effects of Interruption at Different Moments within Task Execution. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vienna, Austria) *(CHI '04)*. Association for Computing Machinery, New York, NY, USA, 271–278. https://doi.org/10.1145/985692.985727

[2] Maryam Alavi. 1994. Computer-Mediated Collaborative Learning: An Empirical Evaluation. *MIS Quarterly* 18, 2 (1994), 159–174. http://www.jstor.org/stable/249763

[3] S. Ali, L. C. Briand, H. Hemmati, and R. K. Panesar-Walawege. 2010. A Systematic Review of the Application and Empirical Investigation of Search-Based Test Case Generation. *IEEE Transactions on Software Engineering* 36, 6 (2010), 742–762.

[4] Ali Alkhatlan and Jugal Kalita. 2018. Intelligent Tutoring Systems: A Comprehensive Historical Survey with Recent Developments. arXiv:1812.09628 [cs.HC]

[5] Teresa M. Amabile and Michael G. Pratt. 2016. The dynamic componential model of creativity and innovation in organizations: Making progress, making meaning. *Research in Organizational Behavior* 36 (2016), 157 – 183. https://doi.org/10.1016/j.riob.2016.10.001

[6] Amazon. 2020. Virtual Assistant Amazon Alexa . https://developer.amazon.com/en-US/alexa

[7] Apple. 2020. Virtual Assistant Apple Siri . https://www.apple.com/siri/

[8] E. Arisholm, H. Gallis, T. Dybå, and D. I. K. Sjoberg. 2007. Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise. *IEEE Transactions on Software Engineering* 33, 2 (2007), 65–86.

[9] Michael Armstrong. 2012. *Armstrong's handbook of reward management practice: Improving performance through reward* (12 ed.). Kogan Page Publishers, London.

[10] Zahra Ashktorab, Mohit Jain, Q. Vera Liao, and Justin D. Weisz. 2019. Resilient Chatbots: Repair Strategy Preferences for Conversational Breakdowns. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, Article 254, 12 pages. https://doi.org/10.1145/3290605.3300484

[11] Prashant Baheti, Edward F. Gehringer, and P. David Stotts. 2002. Exploring the Efficacy of Distributed Pair Programming. In *Proceedings of the Second XP Universe and First Agile Universe Conference on Extreme Programming and Agile Methods - XP/Agile Universe 2002*. Springer-Verlag, Berlin, Heidelberg, 208–220.

[12] Ryan S.J.d. Baker. 2007. Modeling and Understanding Students' off-Task Behavior in Intelligent Tutoring Systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '07)*. Association for Computing Machinery, New York, NY, USA, 1059–1068. https://doi.org/10.1145/1240624.1240785

[13] A. Bandura. 1986. *Social Foundations of Thought and Action: A Social Cognitive Theory*. Prentice-Hall, Michigan. https://books.google.com/books?id=HJhqAAAAMAAJ

[14] Alejandro Barredo Arrieta, Natalia DÃaz-RodrÃguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion* 58 (2020), 82 – 115. http://www.sciencedirect.com/science/article/pii/S1566253519308103

[15] A. Belshee. 2005. Promiscuous pairing and beginner's mind: embrace inexperience [agile programming]. In *Agile Development Conference (ADC'05)*. Agile Development Conference, Denver, Colorado, 125–131. https://doi.org/10.1109/ADC.2005.37

[16] Jeannette Bennett, Kendra Cooper, and Lirong Dai. 2010. Aspect-oriented model-driven skeleton code generation: A graph-based transformation approach. *Science of Computer Programming* 75, 8 (2010), 689–725.

[17] Timothy Bickmore and Justine Cassell. 2001. Relational Agents: A Model and Implementation of Building User Trust. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Seattle, Washington, USA) *(CHI '01)*. ACM, New York, NY, USA, 396–403. https://doi.org/10.1145/365024.365304

[18] Jay Bradley, David Benyon, Oli Mival, and Nick Webb. 2010. Wizard of Oz Experiments and Companion Dialogues. In *Proceedings of the 24th BCS Interaction Specialist Group Conference* (Dundee, United Kingdom) *(BCS '10)*. BCS Learning & Development Ltd., Swindon, GBR, 117–123.

[19] Sheryl Brahnam and Antonella De Angeli. 2012. Gender affordances of conversational agents. *Interacting with Computers* 24, 3 (04 2012), 139–153.

https://doi.org/10.1016/j.intcom.2012.05.001
[20] Virginia Braun and Victoria Clarke. 2006. Using Thematic Analysis in Psychology. *Qualitative research in psychology* 3 (01 2006), 77–101. https://doi.org/10.1191/1478088706qp063oa
[21] Tim Brown. 2009. *Change by Design: How Design Thinking Transforms Organizations and Inspires Innovation.* HarperBusiness, New York.
[22] Margaret Burnett, Anicia Peters, Charles Hill, and Noha Elarief. 2016. Finding Gender-Inclusiveness Software Issues with GenderMag: A Field Investigation. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, San Jose, 2586–2598.
[23] Margaret Burnett, Simone Stumpf, Jamie Macbeth, Stephann Makri, Laura Beckwith, Irwin Kwan, Anicia Peters, and William Jernigan. 2016. GenderMag: A Method for Evaluating Software's Gender Inclusiveness. *Interacting with Computers* forthcoming (01 2016). https://doi.org/10.1093/iwc/iwv046
[24] Margaret M. Burnett. 2020. GenderMag. http://gendermag.org/
[25] Ramón Burri. 2018. *Improving user trust towards conversational chatbot interfaces with voice output.* Master's thesis. KTH, School of Electrical Engineering and Computer Science (EECS).
[26] Judith Butler. 1999. Revisiting Bodies and Pleasures. *Theory, Culture & Society* 16, 2 (1999), 11–20. https://doi.org/10.1177/02632769922050520
[27] Lan Cao, Kannan Mohan, Peng Xu, and Balasubramaniam Ramesh. 2004. How Extreme Does Extreme Programming Have to Be? Adapting XP Practices to Large-Scale Projects. In *Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 3 - Volume 3 (HICSS '04)*. IEEE Computer Society, USA, 30083.3.
[28] Robert Cartwright, Eric Allen, and Charles Reis. 2002. Production Programming in the Classroom. *ACM SIGCSE Bulletin* 35 (11 2002). https://doi.org/10.1145/611892.611940
[29] Mehmet Celepkolu and Kristy Elizabeth Boyer. 2018. The Importance of Producing Shared Code Through Pair Programming. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (Baltimore, Maryland, USA) *(SIGCSE '18)*. Association for Computing Machinery, New York, NY, USA, 765–770. https://doi.org/10.1145/3159450.3159506
[30] Mehmet Celepkolu and Kristy Elizabeth Boyer. 2018. Thematic Analysis of Students' Reflections on Pair Programming in CS1. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (Baltimore, Maryland, USA) *(SIGCSE '18)*. Association for Computing Machinery, New York, NY, USA, 771–776. https://doi.org/10.1145/3159450.3159516
[31] Christopher P Cerasoli, Jessica M Nicklin, and Michael T Ford. 2014. Intrinsic motivation and extrinsic incentives jointly predict performance: A 40-year meta-analysis. *Psychological bulletin* 140, 4 (2014), 980.
[32] Ruth W Chabay and Jill H Larkin. 2020. *Computer assisted instruction and intelligent tutoring systems: Shared goals and complementary approaches.* Routledge, Abingdon, United Kingdom.
[33] Gary Charness and Uri Gneezy. 2012. Strong Evidence for Gender Differences in Risk Taking. *Journal of Economic Behavior & Organization* 83, 1 (2012), 50–58.
[34] J. Y. C. Chen and M. J. Barnes. 2014. Human–Agent Teaming for Multirobot Control: A Review of Human Factors Issues. *IEEE Transactions on Human-Machine Systems* 44, 1 (2014), 13–29.
[35] K. S. Choi. 2013. Evaluating Gender Significance within a Pair Programming Context. In *2013 46th Hawaii International Conference on System Sciences*. IEEE, Hawaii, 4817–4825. https://doi.org/10.1109/HICSS.2013.209
[36] Jan Chong and Tom Hurlbutt. 2007. The Social Dynamics of Pair Programming. In *Proceedings of the 29th International Conference on Software Engineering (ICSE '07)*. IEEE Computer Society, USA, 354–363. https://doi.org/10.1109/ICSE.2007.87
[37] Alistair Cockburn and Laurie Williams. 2001. *The Costs and Benefits of Pair Programming.* Addison-Wesley Longman Publishing Co., Inc., USA, 223–243.
[38] Deborah R. Compeau and Christopher A. Higgins. 1995. Computer Self-Efficacy: Development of a Measure and Initial Test. *MIS Q.* 19, 2 (June 1995), 189–211. https://doi.org/10.2307/249688
[39] Pedro Costa and Luísa Ribas. 2019. AI becomes her: Discussing gender and artificial intelligence. *Technoetic Arts: A Journal of Speculative Research* 17, 1/2 (2019), 171 – 193.
[40] Sally Jo Cunningham, Annika Hinze, and David M. Nichols. 2016. Supporting Gender-Neutral Digital Library Creation: A Case Study Using the GenderMag Toolkit. In *Digital Libraries: Knowledge, Information, and Data in an Open Access Society*, Atsuyuki Morishima, Andreas Rauber, and Chern Li Liew (Eds.). Springer International Publishing, Cham, 45–50.
[41] Nils Dahlbäck, Arne Jönsson, and Lars Ahrenberg. 1993. Wizard of Oz studies—why and how. *Knowledge-based systems* 6, 4 (1993), 258–266.
[42] M. Day, M. R. Penumala, and J. Gonzalez-Sanchez. 2019. Annete: An Intelligent Tutoring Companion Embedded into the Eclipse IDE. In *2019 IEEE First International Conference on Cognitive Machine Intelligence (CogMI)*. IEEE, New York, US, 71–80.
[43] Claudio León de la Barra and Broderick Crawford. 2007. Fostering Creativity Thinking in Agile Software Development. In *HCI and Usability for Medicine and Health Care*, Andreas Holzinger (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 415–426.
[44] Angel de Vicente and Helen Pain. 1998. Motivation Diagnosis in Intelligent Tutoring Systems. In *Intelligent Tutoring Systems*, Barry P. Goettl, Henry M. Halff, Carol L. Redfield, and Valerie J. Shute (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 86–95.
[45] Edward L Deci, Anja H Olafsen, and Richard M Ryan. 2017. Self-determination theory in work organizations: The state of a science. *Annual Review of Organizational Psychology and Organizational Behavior* 4 (2017), 19–43.
[46] Doris M. Dehn and Susanne van Mulken. 2000. The Impact of Animated Interface Agents: A Review of Empirical Research. *Int. J. Hum.-Comput. Stud.* 52, 1 (Jan. 2000), 1–22. https://doi.org/10.1006/ijhc.1999.0325
[47] Tom DeMarco and Tim Lister. 2013. *Peopleware: Productive Projects and Teams (3rd Edition)* (3rd ed.). Addison-Wesley Professional, Boston, MA, USA.
[48] Melissa C. Duffy and Roger Azevedo. 2015. Motivation matters: Interactions between achievement goals and agent scaffolding for self-regulated learning within an intelligent tutoring system. *Computers in Human Behavior* 52 (2015), 338 – 348. http://www.sciencedirect.com/science/article/pii/S0747563215004227
[49] Rafael Duque and Crescencio Bravo. 2008. Analyzing Work Productivity and Program Quality in Collaborative Programming. In *Proceedings of the 2008 The Third International Conference on Software Engineering Advances (ICSEA '08)*. IEEE Computer Society, Washington, DC, USA, 270–276. https://doi.org/10.1109/ICSEA.2008.82
[50] Tore Dybå, Erik Arisholm, Dag Sjøberg, Jo Hannay, and Forrest Shull. 2007. Are Two Heads Better than One? On the Effectiveness of Pair Programming. *Software, IEEE* 24 (12 2007), 12 – 15. https://doi.org/10.1109/MS.2007.158
[51] Berland Edelman and Inc. 2010. *Creativity and education: Why it matters.* Adobe. Retrieved September 18th, 2019 from http://www.adobe.com/aboutadobe/pressroom/pdfs/Adobe_Creativity_and_Education_Why_It_Matters_study.pdf
[52] Katrina Falkner, Nickolas J.G. Falkner, and Rebecca Vivian. 2013. Collaborative Learning and Anxiety: A Phenomenographic Study of Collaborative Learning Activities. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* (Denver, Colorado, USA) *(SIGCSE '13)*. Association for Computing Machinery, New York, NY, USA, 227–232. https://doi.org/10.1145/2445196.2445268
[53] Carmen Fischer, Charlotte P. Malycha, and Ernestine Schafmann. 2019. The Influence of Intrinsic Motivation and Synergistic Extrinsic Motivators on Creativity and Innovation. *Frontiers in Psychology* 10 (2019), 137. https://doi.org/10.3389/fpsyg.2019.00137
[54] S.T. Fiske, E.H.P.P.S.T. Fiske, and S.E. Taylor. 1991. *Social Cognition.* McGraw-Hill, New York City, USA. https://books.google.com/books?id=6Uq3QgAACAAJ
[55] H. Gallis, Erik Arisholm, and Tore Dybå. 2002. A Transition From Partner Programming to Pair Programming - an Industrial Case Study. In *Workshop "Pair Programming Installed" in 17th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA), Position paper.* ACM, Seattle USA, –.
[56] H. Gallis, E. Arisholm, and T. Dyba. 2003. An initial framework for research on pair programming. In *2003 International Symposium on Empirical Software Engineering, 2003. ISESE 2003. Proceedings.* ACM, NY, USA, 132–142. https://doi.org/10.1109/ISESE.2003.1237972
[57] H. Gallis, E. Arisholm, and T. Dyba. 2003. An initial framework for research on pair programming. In *2003 International Symposium on Empirical Software Engineering*. ACM, NY, USA, 132–142.
[58] Alex Gerdes, Bastiaan Heeren, Johan Jeuring, and L. Thomas van Binsbergen. 2016. Ask-Elle: an Adaptable Programming Tutor for Haskell Giving Automated Feedback. *International Journal of Artificial Intelligence in Education* 27 (02 2016). https://doi.org/10.1007/s40593-015-0080-x
[59] Nadine Glas and Catherine Pelachaud. 2015. Definitions of Engagement in Human-Agent Interaction. In *International Conference on Affective Computing and Intelligent Interaction (ACII)*. IEEE, USA, 944–949. https://doi.org/10.1109/ACII.2015.7344688
[60] Barney G. Glaser and Anselm L. Strauss. 1967. *The Discovery of Grounded Theory: Strategies for Qualitative Research.* Aldine de Gruyter, New York, NY.
[61] Bradley Goodman, Amy Soller, Frank Linton, Robert Gaimari, and The Mitre. 1998. Encouraging student reflection and articulation using a learning companion. *International Journal of Artificial Intelligence in Education* 9 (1998), 237–255.
[62] Google. 2019. Google text-to-speech python library. https://github.com/pndurette/gTTS
[63] Google. 2020. Virtual Assistant Google Assistant . https://assistant.google.com/
[64] Jonathan Gratch, Ning Wang, Jillian Gerten, Edward Fast, and Robin Duffy. 2007. Creating Rapport with Virtual Agents. In *Intelligent Virtual Agents*, Catherine Pelachaud, Jean-Claude Martin, Elisabeth André, Gérard Chollet, Kostas Karpouzis, and Danielle Pelé (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 125–138.
[65] J.P. Guilford. 1968. *Intelligence, Creativity, and Their Educational Implications.* R. R. Knapp, Open Library. https://books.google.com/books?id=WE8kAQAAMAAJ

[66] Keun-Woo Han, EunKyoung Lee, and Youngjun Lee. 2010. The Impact of a Peer-Learning Agent Based on Pair Programming in a Programming Course. *Education, IEEE Transactions on* 53 (06 2010), 318 – 327. https://doi.org/10.1109/TE.2009.2019121

[67] Brian Hanks. 2008. Empirical evaluation of distributed pair programming. *International Journal of Human-Computer Studies* 66 (07 2008), 530–544. https://doi.org/10.1016/j.ijhcs.2007.10.003

[68] Brian F. Hanks. 2004. Distributed Pair Programming: An Empirical Study. In *Extreme Programming and Agile Methods - XP/Agile Universe 2004*, Carmen Zannier, Hakan Erdogmus, and Lowell Lindstrom (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 81–91.

[69] Dai Hasegawa, Justine Cassell, and Kenji Araki. 2010. The Role of Embodiment and Perspective in Direction-Giving Systems. In *Proceedings of AAAI Fall Workshop on Dialog with Robots*. AAAI PRESS, USA.

[70] Renate Häuslschmid, Max von Bülow, Bastian Pfleging, and Andreas Butz. 2017. SupportingTrust in Autonomous Driving. In *Proceedings of the 22Nd International Conference on Intelligent User Interfaces* (Limassol, Cyprus) *(IUI '17)*. ACM, New York, NY, USA, 319–329. https://doi.org/10.1145/3025171.3025198

[71] Charles G. Hill, Maren Haag, Alannah Oleson, Chris Mendez, Nicola Marsden, Anita Sarma, and Margaret Burnett. 2017. Gender-Inclusiveness Personas vs. Stereotyping: Can We Have It Both Ways?. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 6658–6671. https://doi.org/10.1145/3025453.3025609

[72] Anthony J. Hillesheim, Christina F. Rusnock, Jason M. Bindewald, and Michael E. Miller. 2017. Relationships between User Demographics and User Trust in an Autonomous Agent. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 61, 1 (2017), 314–318. https://doi.org/10.1177/1541931213601560

[73] Kevin Anthony Hoff and Masooda Bashir. 2015. Trust in Automation: Integrating Empirical Evidence on Factors That Influence Trust. *Human Factors* 57, 3 (2015), 407–434. https://doi.org/10.1177/0018720814547570 PMID: 25875432.

[74] IBM. 2019. Eclipse IDE. https://www.eclipse.org/

[75] Paul Jaccard. 1901. Etude de la distribution florale dans une portion des Alpes et du Jura. *Bulletin de la Societe Vaudoise des Sciences Naturelles* 37 (01 1901), 547–579. https://doi.org/10.5169/seals-266450

[76] Mohit Jain, Pratyush Kumar, Ishita Bhansali, Q. Vera Liao, Khai Truong, and Shwetak Patel. 2018. FarmChat: A Conversational Agent to Answer Farmer Queries. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 4, Article 170 (Dec. 2018), 22 pages. https://doi.org/10.1145/3287048

[77] Mohit Jain, Pratyush Kumar, Ramachandra Kota, and Shwetak N. Patel. 2018. Evaluating and Informing the Design of Chatbots. In *Proceedings of the 2018 Designing Interactive Systems Conference* (Hong Kong, China) *(DIS '18)*. Association for Computing Machinery, New York, NY, USA, 895–906. https://doi.org/10.1145/3196709.3196735

[78] D. James and S. Clarke. 1998. Women, men, and interruptions: A critical review. In *D. Tannen (Ed.), Oxford studies in sociolinguistics. Gender and conversational interaction*. Oxford University Press, UK, 231–280.

[79] Lindsay Jarratt, Nicholas A. Bowman, K.C. Culver, and Alberto Maria Segre. 2019. A Large-Scale Experimental Study of Gender and Pair Composition in Pair Programming. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education* (Aberdeen, Scotland Uk) *(ITiCSE '19)*. Association for Computing Machinery, New York, NY, USA, 176–181. https://doi.org/10.1145/3304221.3319782

[80] Will Jerigan, Amber Horvath, Michael Lee, Margaret Burnett, Cuilty Taylor, Sandeep Kuttal, Anicia Peters, Irwin Kwan, Faezeh Bahmani, and Andrew Ko. 2015. A Principled Evaluation for a Principled Idea Garden. In *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, USA. https://doi.org/10.1109/VLHCC.2015.7357222

[81] William Jernigan, Amber Horvath, Michael Lee, Margaret Burnett, Taylor Cuilty, Sandeep Kuttal, Anicia Peters, Irwin Kwan, Faezeh Bahmani, Andrew Ko, Christopher J. Mendez, and Alannah Oleson. 2017. General principles for a Generalized Idea. *Journal of Visual Languages & Computing* 39 (2017), 51 – 65. https://doi.org/10.1016/j.jvlc.2017.04.005 Special Issue on Programming and Modelling Tools.

[82] Xiaoming Jiang and Marc D. Pell. 2015. On how the brain decodes vocal cues about speaker confidence. *Cortex* 66 (2015), 9 – 34. http://www.sciencedirect.com/science/article/pii/S0010945215000593

[83] Ewa Kacewicz, James W. Pennebaker, Matthew Davis, Moongee Jeon, and Arthur C. Graesser. 2014. Pronoun Use Reflects Standings in Social Hierarchies. *Journal of Language and Social Psychology* 33, 2 (2014), 125–143. https://doi.org/10.1177/0261927X13502654

[84] Peter H. Kahn, Nathan G. Freier, Takayuki Kanda, Hiroshi Ishiguro, Jolina H. Ruckert, Rachel L. Severson, and Shaun K. Kane. 2008. Design Patterns for Sociality in Human-Robot Interaction. In *Proceedings of the 3rd ACM/IEEE International Conference on Human Robot Interaction* (Amsterdam, The Netherlands) *(HRI '08)*. Association for Computing Machinery, New York, NY, USA, 97–104. https://doi.org/10.1145/1349822.1349836

[85] Alexander Karan, Robert Rosenthal, and Megan L. Robbins. 2019. Meta-analytic evidence that we-talk predicts relationship and personal functioning in romantic couples. *Journal of Social and Personal Relationships* 36, 9 (2019), 2624–2651. https://doi.org/10.1177/0265407518795336

[86] Neha Katira, Laurie Williams, Eric Wiebe, Carol Miller, Suzanne Balik, and Ed Gehringer. 2004. On Understanding Compatibility of Student Pair Programmers. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education* (Norfolk, Virginia, USA) *(SIGCSE '04)*. Association for Computing Machinery, New York, NY, USA, 7–11. https://doi.org/10.1145/971300.971307

[87] S. Kaur Kuttal, K. Gerstner, and A. Bejarano. 2019. 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)*. IEEE, USA, 75–85.

[88] Iman Keivanloo, Juergen Rilling, and Ying Zou. 2014. Spotting Working Code Examples. In *Proceedings of the 36th International Conference on Software Engineering* (Hyderabad, India) *(ICSE 2014)*. Association for Computing Machinery, New York, NY, USA, 664–675. https://doi.org/10.1145/2568225.2568292

[89] Kisub Kim, Dongsun Kim, Tegawendé F. Bissyandé, Eunjong Choi, Li Li, Jacques Klein, and Yves Le Traon. 2018. FaCoY: A Code-to-Code Search Engine. In *Proceedings of the 40th International Conference on Software Engineering* (Gothenburg, Sweden) *(ICSE '18)*. Association for Computing Machinery, New York, NY, USA, 946–957. https://doi.org/10.1145/3180155.3180187

[90] Barbara Kitchenham, S.L. Pfleeger, L.M. Pickard, Peter Jones, David Hoaglin, Khaled Emam, and Jarrett Rosenberg. 2002. Preliminary Guidelines for Empirical Research in Software Engineering. *Software Engineering, IEEE Transactions on* 28 (09 2002), 721– 734. https://doi.org/10.1109/TSE.2002.1027796

[91] Dominique Knutsen and Ludovic Le Bigot. 2014. The Influence of Reference Acceptance and Reuse on Conversational Memory Traces. *Journal of experimental psychology. Learning, memory, and cognition* 41 (07 2014). https://doi.org/10.1037/xlm0000036

[92] Dominique Knutsen, Ludovic Le Bigot, and Christine Ros. 2017. Explicit feedback from users attenuates memory biases in human-system dialogue. *International Journal of Human-Computer Studies* 97 (2017), 77 – 87. http://www.sciencedirect.com/science/article/pii/S1071581916301045

[93] Dominique Knutsen, Christine Ros, and Ludovic Le Bigot. 2016. Generating References in Naturalistic Face-to-Face and Phone-Mediated Dialog Settings. *Topics in Cognitive Science* 8 (08 2016). https://doi.org/10.1111/tops.12218

[94] Andrew J. Ko and Brad A. Myers. 2004. Designing the Whyline: A Debugging Interface for Asking Questions about Program Behavior. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vienna, Austria) *(CHI '04)*. Association for Computing Machinery, New York, NY, USA, 151–158. https://doi.org/10.1145/985692.985712

[95] Andrew J. Ko and Brad A. Myers. 2008. Debugging Reinvented: Asking and Answering Why and Why Not Questions about Program Behavior. In *Proceedings of the 30th International Conference on Software Engineering* (Leipzig, Germany) *(ICSE '08)*. Association for Computing Machinery, New York, NY, USA, 301–310. https://doi.org/10.1145/1368088.1368130

[96] Kuttal, Kwasny, Ong, and Robe. 2020. Correctness and Progress Metrics . https://docs.google.com/spreadsheets/d/1UawMLVACqLjC6JH7I5vwWVQLCv6phlkCllZDwHkM3Y/edit?usp=sharing

[97] S. K. Kuttal, J. Myers, S. Gurka, D. Magar, D. Piorkowski, and R. Bellamy. 2020. Towards Designing Conversational Agents for Pair Programming: Accounting for Creativity Strategies and Conversational Styles. In *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, USA, 1–11.

[98] Danielle L. Jones and Scott D. Fleming. 2013. What use is a backseat driver? A qualitative investigation of pair programming. In *Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC*. IEEE, USA, 103–110.

[99] Thomas K Landauer. 1987. Psychology as a mother of invention. *ACM SIGCHI Bulletin* 18, 4 (1987), 333–335.

[100] Susan Leavy. 2018. Gender Bias in Artificial Intelligence: The Need for Diversity and Gender Theory in Machine Learning. In *Proceedings of the 1st International Workshop on Gender Equality in Software Engineering* (Gothenburg, Sweden) *(GE '18)*. Association for Computing Machinery, New York, NY, USA, 14–16. https://doi.org/10.1145/3195570.3195580

[101] Susan Leavy. 2018. Gender Bias in Artificial Intelligence: The Need for Diversity and Gender Theory in Machine Learning. In *Proceedings of the 1st International Workshop on Gender Equality in Software Engineering* (Gothenburg, Sweden) *(GE '18)*. Association for Computing Machinery, New York, NY, USA, 14–16. https://doi.org/10.1145/3195570.3195580

[102] Marvin Levine. 1988. *Effective problem solving*. Prentice Hall, NJ, USA.

[103] Clayton Lewis. 1982. *Using the "thinking-aloud" method in cognitive interface design*. IBM T.J. Watson Research Center, Yorktown Heights, N.Y.

[104] Colleen M. Lewis and Niral Shah. 2015. How Equity and Inequity Can Emerge in Pair Programming. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research* (Omaha, Nebraska, USA) *(ICER '15)*. Association for Computing Machinery, New York, NY, USA, 41–50. https://doi.org/10.1145/2787622.2787716

[105] Zhen Li and Eileen Kraemer. 2014. Social Effects of Pair Programming Mitigate Impact of Bounded Rationality. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (Atlanta, Georgia, USA) *(SIGCSE '14)*. Association for Computing Machinery, New York, NY, USA, 385–390. https://doi.org/10.1145/2538862.2538968

[106] Dapeng Liu, Andrian Marcus, Denys Poshyvanyk, and Vaclav Rajlich. 2007. Feature Location via Information Retrieval Based Filtering of a Single Scenario Execution Trace. In *Proceedings of the Twenty-Second IEEE/ACM International Conference on Automated Software Engineering* (Atlanta, Georgia, USA) *(ASE '07)*. Association for Computing Machinery, New York, NY, USA, 234–243. https://doi.org/10.1145/1321631.1321667

[107] Zhiqiang Liu and Dieter J Schonwetter. 2004. Teaching creativity in engineering. *International Journal of Engineering Education* 20, 5 (2004), 801–808.

[108] Irene Lopatovska and Harriet Williams. 2018. Personification of the Amazon Alexa: BFF or a Mindless Companion. In *Proceedings of the 2018 Conference on Human Information Interaction & Retrieval* (New Brunswick, NJ, USA) *(CHIIR '18)*. Association for Computing Machinery, New York, NY, USA, 265–268. https://doi.org/10.1145/3176349.3176868

[109] Ewa Luger and Abigail Sellen. 2016. "Like Having a Really Bad PA": The Gulf between User Expectation and Experience of Conversational Agents. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 5286–5297. https://doi.org/10.1145/2858036.2858288

[110] A. Marcus, V. Rajlich, J. Buchta, M. Petrenko, and A. Sergeyev. 2005. Static techniques for concept location in object-oriented code. In *13th International Workshop on Program Comprehension (IWPC'05)*. IEEE Computer Society, Los Alamitos, CA, USA, 33–42.

[111] Yukihiro Matsubara and Mitsuo Nagamachi. 1996. Motivation system and human model for intelligent tutoring. In *Intelligent Tutoring Systems*, Claude Frasson, Gilles Gauthier, and Alan Lesgold (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 139–147.

[112] Charlie McDowell, Linda Werner, Heather Bullock, and Julian Fernald. 2002. The Effects of Pair-Programming on Performance in an Introductory Programming Course. In *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education* (Cincinnati, Kentucky) *(SIGCSE '02)*. Association for Computing Machinery, New York, NY, USA, 38–42. https://doi.org/10.1145/563340.563353

[113] Charles Mcdowell, Linda Werner, Heather Bullock, and Julian Fernald. 2006. Pair programming improves student retention, confidence, and program quality. *Commun. ACM* 49 (08 2006), 90–95. https://doi.org/10.1145/1145293

[114] Charlie McDowell, Linda Werner, Heather E. Bullock, and Julian Fernald. 2003. The Impact of Pair Programming on Student Performance, Perception and Persistence. In *Proceedings of the 25th International Conference on Software Engineering* (Portland, Oregon) *(ICSE '03)*. IEEE Computer Society, USA, 602–607.

[115] Meiliana, Irwandhi Septian, Ricky Setiawan Alianto, Daniel, and Ford Lumban Gaol. 2017. Automated Test Case Generation from UML Activity Diagram and Sequence Diagram using Depth First Search Algorithm. *Procedia Computer Science* 116 (2017), 629 – 637. http://www.sciencedirect.com/science/article/pii/S1877050917320732 Discovery and innovation of computer science technology in artificial intelligence era: The 2nd International Conference on Computer Science and Computational Intelligence (ICCSCI 2017).

[116] Christopher Mendez, Hema Susmita Padala, Zoe Steine-Hanson, Claudia Hilderbrand, Amber Horvath, Charles Hill, Logan Simpson, Nupoor Patil, Anita Sarma, and Margaret Burnett. 2018. Open Source Barriers to Entry, Revisited: A Sociotechnical Perspective. In *Proceedings of the 40th International Conference on Software Engineering* (Gothenburg, Sweden) *(ICSE '18)*. Association for Computing Machinery, New York, NY, USA, 1004–1015. https://doi.org/10.1145/3180155.3180241

[117] M. Minsky, R. Kurzweil, and S. Mann. 2013. The society of intelligent veillance. In *2013 IEEE International Symposium on Technology and Society (ISTAS): Social Implications of Wearable Computing and Augmediated Reality in Everyday Life*. IEEE, USA, 13–17.

[118] Matheus Monteiro, Erica Souza, Andre Endo, and Nandamudi Vijaykumar. 2019. Analyzing graph-based algorithms employed to generate test cases from finite state machines. In *2019 IEEE Latin American Test Symposium (LATS)*. IEEE, USA. https://doi.org/10.1109/LATW.2019.8704603

[119] Susanne van Mulken, Elisabeth André, and Jochen Müller. 1999. An Empirical Study on the Trustworthiness of Life-like Interface Agents. In *Proceedings of the HCI International '99 (the 8th International Conference on Human-Computer Interaction) on Human-Computer Interaction: Communication, Cooperation, and Application Design-Volume 2 - Volume 2*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 152–156.

[120] Kumiko Murata. 1994. Intrusive or co-operative? A cross-cultural study of interruption. *Journal of Pragmatics* 21, 4 (1994), 385 – 400. http://www.sciencedirect.com/science/article/pii/0378216694900116

[121] Emerson Murphy-Hill and Gail C. Murphy. 2011. Peer Interaction Effectively, yet Infrequently, Enables Programmers to Discover New Tools. In *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work* (Hangzhou, China) *(CSCW '11)*. Association for Computing Machinery, New York, NY, USA, 405–414. https://doi.org/10.1145/1958824.1958888

[122] Nachiappan Nagappan, Laurie Williams, Miriam Ferzli, Eric Wiebe, Kai Yang, Carol Miller, and Suzanne Balik. 2003. Improving the CS1 Experience with Pair Programming. *SIGCSE Bull.* 35, 1 (Jan. 2003), 359–362. https://doi.org/10.1145/792548.612006

[123] Sik Hung Ng, Mark Brooke, and Michael Dunne. 1995. Interruption and Influence in Discussion Groups. *Journal of Language and Social Psychology* 14, 4 (1995), 369–381. https://doi.org/10.1177/0261927X950144003

[124] Haoran Niu, Iman Keivanloo, and Ying Zou. 2017. Learning to rank code examples for code search engines. *Empirical Software Engineering* 22, 1 (2017), 259–291.

[125] John Nosek. 1998. The Case for Collaborative Programming. *Commun. ACM* 41 (03 1998). https://doi.org/10.1145/272287.272333

[126] Clem O'Donnell, Jim Buckley, Abdulhussain Mahdi, John Nelson, and Michael English. 2015. Evaluating Pair-Programming for Non-Computer Science Major Students. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (Kansas City, Missouri, USA) *(SIGCSE '15)*. Association for Computing Machinery, New York, NY, USA, 569–574. https://doi.org/10.1145/2676723.2677289

[127] Sharon Oviatt and Philip Cohen. 2000. Perceptual User Interfaces: Multimodal Interfaces That Process What Comes Naturally. *Commun. ACM* 43, 3 (March 2000), 45–53. https://doi.org/10.1145/330534.330538

[128] Frank Pajares. 2002. Overview of social cognitive theory and of self-efficacy.

[129] David Walsh Palmieri. 2002. Knowledge Management Through Pair Programming.

[130] George Polya. 2004. *How to solve it: A new aspect of mathematical method*. Vol. 85. Princeton university press, NJ, USA.

[131] Leo Porter and Beth Simon. 2013. Retaining Nearly One-Third More Majors with a Trio of Instructional Best Practices in CS1. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* (Denver, Colorado, USA) *(SIGCSE '13)*. Association for Computing Machinery, New York, NY, USA, 165–170. https://doi.org/10.1145/2445196.2445248

[132] Lutz Prechelt, Ulrich Stärk, and Stephan Salinger. 2009. 7 Types of Cooperation Episodes in Side-by-Side Programming. In *Proc. 21st Annual Workshop of the Psychology of Programming Interest Group (PPIG '09)*. ACM, USA.

[133] Mukund Raghothaman, Yi Wei, and Youssef Hamadi. 2016. SWIM: Synthesizing What i Mean: Code Search and Idiomatic Snippet Synthesis. In *Proceedings of the 38th International Conference on Software Engineering* (Austin, Texas) *(ICSE '16)*. Association for Computing Machinery, New York, NY, USA, 357–367. https://doi.org/10.1145/2884781.2884808

[134] P. Rane. 2017. Automatic Generation of Test Cases for Agile using Natural Language Processing.

[135] Genaro Rebolledo-Mendez, Sara de Freitas, Jose Rafael Rojano-Caceres, and Alma Rosa García-Gaona. 2010. An Empirical Examination of the Relation Between Attention and Motivation in Computer-Based Education: a Modeling Approach. In *Proceedings of the Twenty-Third International Florida Artificial Intelligence Research Society Conference, May 19-21, 2010, Daytona Beach, Florida, USA*, Hans W. Guesgen and R. Charles Murray (Eds.). AAAI Press, USA, 74–79.

[136] Genaro Rebolledo-Mendez, Benedict du Boulay, and Rosemary Luckin. 2006. Motivating the Learner: An Empirical Evaluation. In *Intelligent Tutoring Systems*, Mitsuru Ikeda, Kevin D. Ashley, and Tak-Wai Chan (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 545–554.

[137] Laurel D Riek. 2012. Wizard of oz studies in hri: a systematic review and new reporting guidelines. *Journal of Human-Robot Interaction* 1, 1 (2012), 119–136.

[138] Peter Robe. 2020. Designing PairBuddy – Conversational Agent for Pair. https://drive.google.com/drive/folders/1vIOdro0pg8C1jSB42KzYrDRKO0PVhqZ1?usp=sharing

[139] T. J. Robertson, Shrinu Prabhakararao, Margaret Burnett, Curtis Cook, Joseph R. Ruthruff, Laura Beckwith, and Amit Phalgune. 2004. Impact of Interruption Style on End-User Debugging. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vienna, Austria) *(CHI '04)*. Association for Computing Machinery, New York, NY, USA, 287–294. https://doi.org/10.1145/985692.985729

[140] Fernando J. Rodríguez, Kimberly Michelle Price, and Kristy Elizabeth Boyer. 2017. Exploring the Pair Programming Process: Characteristics of Effective Collaboration. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (Seattle, Washington, USA) *(SIGCSE '17)*. Association for Computing Machinery, New York, NY, USA, 507–512. https://doi.org/10.1145/3017680.3017748

[141] Omar Ruvalcaba, Linda Werner, and Jill Denner. 2016. Observations of Pair Programming: Variations in Collaboration Across Demographic Groups. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (Memphis, Tennessee, USA) *(SIGCSE '16)*. Association for Computing Machinery, New York, NY, USA, 90–95. https://doi.org/10.1145/2839509.2844558

[142] Jeffrey S. Saltz and Ivan Shamshurin. 2019. Exploring pair programming beyond computer science: a case study in its use in data science/data engineering. *International Journal of Higher Education and Sustainability* 2, 4 (2019), 265–278. https://doi.org/10.1504/IJHES.2019.103360

[143] T. Savage, M. Revelle, and D. Poshyvanyk. 2010. FLAT3: feature location and textual tracing tool. In *2010 ACM/IEEE 32nd International Conference on Software Engineering*, Vol. 2. ACM, USA, 255–258.

[144] Philippe G. Schyns, Lucy S. Petro, and Marie L. Smith. 2009. Transmission of Facial Expressions of Emotion Co-Evolved with Their Efficient Decoding in the Brain: Behavioral and Brain Evidence. *PLOS ONE* 4, 5 (05 2009), 1–16. https://doi.org/10.1371/journal.pone.0005625

[145] C. B. Seaman. 1999. "Qualitative Methods in Empirical Studies of Software Engineering". *IEEE Transactions on Software Engineering* 25, 4 (1999), 557–572.

[146] A. Seeger, J. Pfeiffer, and A. Heinzl. 2017. When Do We Need a Human? Anthropomorphic Design and Trustworthiness of Conversational Agents. In *Proceedings of the SixteenthAnnual Pre-ICIS Workshop on HCI Research in MIS*. ACM, USA, 1–5.

[147] Ameneh Shamekhi, Q. Vera Liao, Dakuo Wang, Rachel K. E. Bellamy, and Thomas Erickson. 2018. Face Value? Exploring the Effects of Embodiment for a Group Facilitation Agent. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3173965

[148] R. Sharma, S. Gulia, and K. K. Biswas. 2014. Automated generation of activity and sequence diagrams from natural language requirements. In *9th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*. SCITEPRESS Digital Library, Portugal, 1–9.

[149] Arun Shekhar and Nicola Marsden. 2018. Cognitive Walkthrough of a Learning Management System with Gendered Personas. In *Proceedings of the 4th Conference on Gender & IT* (Heilbronn, Germany) *(GenderIT '18)*. Association for Computing Machinery, New York, NY, USA, 191–198. https://doi.org/10.1145/3196839.3196869

[150] F. Shull, J. Singer, and D. I. K. Sjøberg. 2008. Guide to Advanced Empirical Software Engineering: Springer.

[151] Candace L. Sidner, Christopher Lee, Cory D. Kidd, Neal Lesh, and Charles Rich. 2005. Explorations in engagement for humans and robots. *Artificial Intelligence* 166, 1 (2005), 140 – 164. http://www.sciencedirect.com/science/article/pii/S0004370205000512

[152] Simon and Susan Snowdon. 2011. Explaining Program Code: Giving Students the Answer Helps - but Only Just. In *Proceedings of the Seventh International Workshop on Computing Education Research* (Providence, Rhode Island, USA) *(ICER '11)*. Association for Computing Machinery, New York, NY, USA, 93–100. https://doi.org/10.1145/2016911.2016931

[153] Zach Sims. 2020. Code Academy. https://www.codecademy.com/

[154] Arsenij Solovjev. 2020. Saros Project. https://www.saros-project.org/

[155] Jörg Spieler. 2020. *UCDetector*. Open Source. http://www.ucdetector.org/

[156] Hugo Spiers, Bradley Love, Mike Pelley, Charlotte Gibb, and Robin Murphy. 2016. Anterior Temporal Lobe Tracks the Formation of Prejudice. *Journal of Cognitive Neuroscience* 29 (10 2016), 1–15. https://doi.org/10.1162/jocn_a_01056

[157] Lee Sproull, Mani Subramani, Sara Kiesler, Janet H. Walker, and Keith Waters. 1996. When the Interface is a Face. *Hum.-Comput. Interact.* 11, 2 (June 1996), 97–124. https://doi.org/10.1207/s15327051hci1102_1

[158] R. Strachan, A. Peixoto, I. Emembolu, and M. T. Restivo. 2018. Women in engineering: Addressing the gender gap, exploring trust and our unconscious bias. In *2018 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, USA, 2088–2093.

[159] Anselm L. Strauss and Juliet M. Corbin. 1998. *Basics of qualitative research: techniques and procedures for developing grounded theory*. Sage Publications, Thousand Oaks, California, USA. XIII, 312 s pages.

[160] Holotech Studios. 2020. Facerig. https://facerig.com/

[161] S. Shyam Sundar and Jinyoung Kim. 2019. Machine Heuristic: When We Trust Computers More than Humans with Our Personal Information. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–9. https://doi.org/10.1145/3290605.3300768

[162] Akikazu Takeuchi and Taketo Naito. 1995. Situated Facial Displays: Towards Social Interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '95)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 450–455. https://doi.org/10.1145/223904.223965

[163] Germany TeamViewer AG. 2019. TeamViewer. https://www.teamviewer.com/en-us/

[164] TechSmith. 2019. Morae. http://www.techsmith.com/morae.asp

[165] Diana-Cezara Toader, Grațiela Boca, Rita Toader, Mara Măcelaru, Cezar Toader, Diana Ighian, and Adrian T. Rădulescu. 2019. The Effect of Social Presence and Chatbot Errors on Trust. *Sustainability* 12, 1 (Dec 2019), 256. https://doi.org/10.3390/su12010256

[166] E. Paul Torrance. 1970. Influence of Dyadic Interaction on Creative Functioning. *Psychological Reports* 26, 2 (1970), 391–394. https://doi.org/10.2466/pr0.1970.26.2.391

[167] Despina Tsompanoudi, Maya Satratzemi, Stelios Xinogalos, and Leonidas Karamitopoulos. 2019. An Empirical Study on Factors related to Distributed Pair Programming. (04 2019). https://www.learntechlib.org/p/208576

[168] Mengping Tsuei. 2017. Learning behaviours of low-achieving children's mathematics learning in using of helping tools in a synchronous peer-tutoring system. *Interactive Learning Environments* 25, 2 (2017), 147–161. https://doi.org/10.1080/10494820.2016.1276078

[169] ALINA Tugend. 2007. Why is asking for help so difficult.

[170] Susanne van Mulken, Elisabeth André, and Jochen Müller. 1998. The Persona Effect: How Substantial Is It?. In *People and Computers XIII*, Hilary Johnson, Lawrence Nigay, and Christopher Roast (Eds.). Springer London, London, 53–66.

[171] Mihaela Vorvoreanu, Lingyi Zhang, Yun-Han Huang, Claudia Hilderbrand, Zoe Steine-Hanson, and Margaret Burnett. 2019. From Gender Biases to Gender-Inclusive Design: An Empirical Investigation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. ACM, New York, NY, USA, Article 53, 14 pages. https://doi.org/10.1145/3290605.3300283

[172] Tony Wagner and Robert A Compton. 2012. *Creating innovators: The making of young people who will change the world*. Simon and Schuster, USA.

[173] P. Wargnier, G. Carletti, Y. Laurent-Corniquet, S. Benveniste, P. Jouvelot, and A. Rigaud. 2016. Field evaluation with cognitively-impaired older adults of attention management in the Embodied Conversational Agent Louise. In *2016 IEEE International Conference on Serious Games and Applications for Health (SeGAH)*. IEEE Computer Society, Los Alamitos, CA, USA, 1–8. https://doi.org/10.1109/SeGAH.2016.7586282

[174] Katharina Weitz, Dominik Schiller, Ruben Schlagowski, Tobias Huber, and Elisabeth André. 2019. "Do You Trust Me?": Increasing User-Trust by Integrating Virtual Agents in Explainable AI Interaction Design. In *Proceedings of the 19th ACM International Conference on Intelligent Virtual Agents* (Paris, France) *(IVA '19)*. Association for Computing Machinery, New York, NY, USA, 7–9. https://doi.org/10.1145/3308532.3329441

[175] Linda L. Werner, Brian Hanks, and Charlie McDowell. 2004. Pair-Programming Helps Female Computer Science Students. *J. Educ. Resour. Comput.* 4, 1 (March 2004), 4–es. https://doi.org/10.1145/1060071.1060075

[176] CANDACE WEST and DON H. ZIMMERMAN. 1987. Doing Gender. *Gender & Society* 1, 2 (1987), 125–151. https://doi.org/10.1177/0891243287001002002

[177] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks. arXiv:1502.05698 [cs.AI]

[178] Wayne A Wickelgren. 1974. *How to solve problems: Elements of a theory of problems and problem solving*. WH Freeman San Francisco, USA.

[179] Laurie Williams and Bob Kessler. 2000. The Effects of "Pair-Pressure" and "Pair-Learning" on Software Engineering Education. In *Proceedings of the 13th Conference on Software Engineering Education & Training (CSEET '00)*. IEEE Computer Society, USA, 59.

[180] Laurie Williams and Robert Kessler. 2002. *Pair Programming Illuminated*. Addison-Wesley Longman Publishing Co., Inc., USA.

[181] L. Williams, R. R. Kessler, W. Cunningham, and R. Jeffries. 2000. Strengthening the case for pair programming. *IEEE Software* 17, 4 (2000), 19–25.

[182] Laurie Williams, D. Scott McCrickard, Lucas Layman, and Khaled Hussein. 2008. Eleven Guidelines for Implementing Pair Programming in the Classroom. In *Proceedings of the Agile 2008 (AGILE '08)*. IEEE Computer Society, USA, 445–452. https://doi.org/10.1109/Agile.2008.12

[183] Laurie Williams, Charlie McDowell, Nachiappan Nagappan, Julian Fernald, and Linda Werner. 2003. Building Pair Programming Knowledge through a Family of Experiments. In *Proceedings of the 2003 International Symposium on Empirical Software Engineering (ISESE '03)*. IEEE Computer Society, USA, 143.

[184] Laurie Williams, Eric Wiebe, Kai Yang, Miriam Ferzli, and Carol Miller. 2002. In Support of Pair Programming in the Introductory Computer Science Course. *Computer Science Education* 12, 3 (2002), 197–212. https://doi.org/10.1076/csed.12.3.197.8618

[185] Laurie A. Williams. 2010. *Pair Programming*. John Wiley & Sons, USA. 311–322 pages.

[186] Laurie A. Williams and Robert R. Kessler. 2000. All I Really Need to Know about Pair Programming I Learned in Kindergarten. *Commun. ACM* 43, 5 (May 2000), 108–114. https://doi.org/10.1145/332833.332848

[187] Aaron Wilson, Margaret Burnett, Laura Beckwith, Orion Granatir, Ledah Casburn, Curtis Cook, Mike Durham, and Gregg Rothermel. 2003. Harnessing Curiosity to Increase Correctness in End-User Programming. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Ft. Lauderdale, Florida, USA) *(CHI '03)*. Association for Computing Machinery, New York, NY, USA, 305–312. https://doi.org/10.1145/642611.642665

[188] James Wilson and Daniel Rosenberg. 1988. Rapid prototyping for user interface design. In *Handbook of human-computer interaction*. Elsevier, Amsterdam, Netherlands, 859–875.

[189] Anita Woolley, Ishani Aggarwal, and Thomas Malone. 2015. Collective Intelligence and Group Performance. *Current Directions in Psychological Science* 24 (12 2015), 420–424. https://doi.org/10.1177/0963721415599543

[190] Nick Yee, Jeremy N Bailenson, and Kathryn Rickertsen. 2007. A Meta-analysis of the Impact of the Inclusion and Realism of Human-like Faces on User Experiences in Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors*

*in Computing Systems* (San Jose, California, USA) *(CHI '07)*. ACM, New York, NY, USA, 1–10. https://doi.org/10.1145/1240624.1240626

[191] Kimberly Michelle Ying, Lydia G. Pezzullo, Mohona Ahmed, Kassandra Crompton, Jeremiah Blanchard, and Kristy Elizabeth Boyer. 2019. In Their Own Words: Gender Differences in Student Perceptions of Pair Programming. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) *(SIGCSE '19)*. Association for Computing Machinery, New York, NY, USA, 1053–1059. https://doi.org/10.1145/3287324.3287380

[192] Jeff Youngquist. 2009. The Effect of Interruptions and Dyad Gender Combination on Perceptions of Interpersonal Dominance. *Communication Studies* 60, 2 (2009), 147–163. https://doi.org/10.1080/10510970902834874

[193] Hans Yuan and Yingjun Cao. 2019. Hybrid Pair Programming - A Promising Alternative to Standard Pair Programming. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) *(SIGCSE '19)*. Association for Computing Machinery, New York, NY, USA, 1046–1052. https://doi.org/10.1145/3287324.3287352

[194] Mohan Zalake, Julia Woodward, Amanpreet Kapoor, and Benjamin Lok. 2018. Assessing the Impact of Virtual Human's Appearance on Users' Trust Levels. In *Proceedings of the 18th International Conference on Intelligent Virtual Agents* (Sydney, NSW, Australia) *(IVA '18)*. Association for Computing Machinery, New York, NY, USA, 329–330. https://doi.org/10.1145/3267851.3267863

[195] Yong Zhao. 2012. *World class learners: Educating creative and entrepreneurial students*. Corwin Press, USA.

[196] Rui Zhi, Samiha Marwan, Yihuan Dong, Nicholas Lytle, Thomas W Price, and Tiffany Barnes. 2019. Toward Data-Driven Example Feedback for Novice Programming. In *International Educational Data Mining Society*. ERIC, USA, 218–227.

[197] Franz Zieris and Lutz Prechelt. 2014. On Knowledge Transfer Skill in Pair Programming. In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (Torino, Italy) *(ESEM '14)*. Association for Computing Machinery, New York, NY, USA, Article 11, 10 pages. https://doi.org/10.1145/2652524.2652529

[198] Franz Zieris and Lutz Prechelt. 2020. Explaining Pair Programming Session Dynamics from Knowledge Gaps. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering* (Seoul, South Korea) *(ICSE '20)*. Association for Computing Machinery, New York, NY, USA, 421–432. https://doi.org/10.1145/3377811.3380925