Designing a Gender-Inclusive Conversational Agent for Pair Programming: An Empirical Investigation

Sandeep Kaur Kuttal[1], Abim Sedhain[2] and Jacob AuBuchon[3]

[1-3] University of Tulsa, Tulsa OK 74104, USA
sandeep-kuttal,abs5423,jsa6790@utulsa.edu

Abstract. Recently, research has shown that replacing a human with an agent in a pair programming context can bring similar benefits such as increased code quality, productivity, self-efficacy, and knowledge transfer as it does with a human. However, to create a gender-inclusive agent, we need to understand the communication styles between human-human and human-agent pairs. To investigate the communication styles, we conducted gender-balanced studies with human-human pairs in a remote lab setting with 18 programmers and human-agent pairs using Wizard-of-Oz methodology with 14 programmers. Our quantitative and qualitative analysis of the communication styles between the two studies showed that humans were more comfortable asking questions to an agent and interacting with it than other humans. We also found men participants showed less uncertainty and trusted agent solutions more, while women participants used more instructions and apologized less to an agent. Our research results confirm the feasibility of creating gender-inclusive conversational agents for programming.

Keywords: Gender, Conversational Agents, Communication Style.

1 Introduction

Conversational agents — such as automated customer support, personal virtual assistants, and social chatbots — have transformed human interactions with computers [40-46]. Despite phenomenal progress in conversational agent's research there does not exist any such agent for programming tasks. To understand the design space of such an agent, we prototyped an interactive pair programming partner agent based on research from conversational agents, software engineering, education, human-robot interactions, psychology, and artificial intelligence [1,3-5].

In pair programming, two programmers work simultaneously on one design, algorithm, code, or test [14-17]. Programmers switch between the roles of driver (writing code) and navigator (making suggestions). Pair programming provides various benefits, including increased code quality, productivity, creativity, knowledge management, and self-efficacy [18-30].

Our agent provided similar benefits as of pair programming with another human[4]. It's active application of social skills as a navigator increased participants' confidence in the code and trust in the agent, while its technical skills as a driver helped participants realize their own solutions better [5].

A key facet in the design of a gender-inclusive conversational agent is how it should account for different communication styles; an area where women and men are known to differ [9-12]. To reduce the possibility of gender biases in our agent, we need to consider communication styles of each gender. Therefore, we formulated the following research questions:

R1. What communication styles are used by programmers when interacting with a human vs. an agent?

R2. How do men's communication styles differ when interacting with a human vs. an agent?

R3. How do women's communication styles differ when interacting with a human vs. an agent?

2 Methodology

A human-human study was conducted in a remote lab setting followed by a human-agent study using Wizard of Oz technique to investigate the similarities and differences between human-human and human-agent interactions.

2.1 Human-Human Study

A human-human study was conducted to analyze interactions between humans in a pair programming environment. 18 computer science students were conducted on a first-come first-serve basis, who each had at least a year of experience and some knowledge in java programming. Based on the background questionnaire, we identified only binary genders (men and women) from their own self-identification, though there are other genders [31, 32]. Hence, we paired students into gender balanced pairs (3 man-man, 3 woman-woman, and 3 man-woman). Gender balanced data is essential for discovering gender biases in designs to avoid unintentional gender bias and to support gender specific problem solving [38, 39], communication techniques, and leadership styles [31-39]. Gender was the focus of the study as opposed to other demographics because of the well-documented negative effects of the gender gap in the STEM fields.

We refer to each pair with the label HH-X with X being the gender of the individual: i.e., HH8-M7 and HH8-W9 refers to the seventh man participant and ninth woman participant who were in pair 8.

Study Design

The study was conducted in a lab setting to emulate a remote pair programming environment. Remote pair programming is known to have benefits in the likeness of collocated pair programming [54-56]. Every participant was required to complete a consent form, background questionnaire, and pre-self-efficacy questionnaire before the study [57]. Participants watched instructional video tutorials to teach them concepts of

test-driven development (e.g., writing test cases, implementing code), pair programming concepts (e.g., driver and navigator roles), and think-aloud study (e.g., vocalize any thoughts and feelings as they program [58]). Each pair communicated remotely using TeamViewer [117] and implementing the Eclipse IDE [118] to complete the task. Before the task was performed, the pairs were given a warm-up task to allow pair jelling, a period that allows them to adjust to their partner and work more efficiently [59].

Participants were given the task of implementing a tic-tac-toe game in Java. The game was selected for the study due to its simplicity and popularity. In tic-tac-toe, two players take turns placing marks in a 3x3 grid until either one player successfully gets three marks consecutively, or winning becomes impossible, causing a tie. Participants needed to write methods and test cases to complete the game; however, methods for a board and the ability to place marks on it were already provided to them. User stories and acceptance criteria for the task were provided regarding win conditions, full board, taking turns, and a tie. Participants determined their own roles as driver and navigator. They were given 40 minutes to complete the task to prevent fatigue. Study sessions were recorded using the Morae screen capture tool [60]. After the session was completed, participants answered questionnaires on post-self-efficacy and their pair programming preferences.

2.2 Human-Agent Study

A Wizard of Oz lab study was conducted to identify the interactions between a human and an agent. The study design, and the data analysis, was like our previous human-human study.

Wizard of Oz

Our study followed the basic components of a Wizard of Oz design. Wizard of Oz design helps to replicate a real virtual agent and effectively identify human interactions with an AI software [61-64]. Two wizards maintained the illusion of the agent, using dialogue options from a templated script. Participants had their face, voice, and screen were shared with the agent (the wizards). The wizards simulated a conversational agent through speech recognition, intent understanding, dialogue state tracking, dialogue policy, and response generation, using a constraint called the Wizard of Oz protocol [66]. For example, if the participant asked, "*How to write code for the win game?*", the wizard would identify this question as "*implementation help*" and would subsequently choose the appropriate response from the wizard's templated script: "*I can make a recommendation from GitHub, would you like me to do so?*"

Participants pair programmed using the Saros plugin for the Eclipse IDE to facilitate remote collaboration with the agent (wizard). The agent (wizard) directly edited participants code using the Saros plugin for Eclipse [77]. Participants interacted with the agent, embodied by a 3D avatar, which was synchronized with the wizard's face using the Facerig software [65]. They communicated directly with the agent using voice and it responded with voice-synthesized messages to reduce switching the context of the participants and to increase interactions with the wizard. The voice-synthesized messages were generated using Google Text-to-Speech. Text communication was used exclusively for sending links and pictures. Skype, Discord, and Google Hangouts were

used to facilitate video and audio communication of the participants with the wizard to give the illusion of a real programming conversational agent.

Agent Design

The agent's design was inspired by multi-disciplinary research on human-computer interactions, conversational agents, human-robotic interactions, education, intelligent tutoring systems, psychology, and management science.

The agent was designed to interact with its partners by using a dynamic 3D avatar, voice, and text chat that enhanced human-computer interactions [3, 67-70]. The inclusion of an avatar makes the agent more human-like, improving understanding, engagement, and trust in novice programmers [61, 71-76].

The agent built rapport with participants as it greeted and introduced itself to them at the beginning of the study [78]. Further, it attributed success to the group and took personal responsibility for its mistakes. To increase participants' trust [67, 79-81] the agent showed uncertainty about its work, asking for verification; for example, after adding code through the IDE, the agent would say "*This might do the trick. I'm not sure though*."

Motivation has a significant effect on performance of programmers and their productivity, also helping them to increase their creativity and their innovative outcomes [82-86]. Motivation was implemented in the study with motivational statements like "*I think this looks good*," and "*I'm not sure what we're doing here, but we can always test it."* being given to participants upon both success and failure.

The agent's ability to contribute code (driver role) and give feedback (navigator role) were based on automated code/feedback techniques that require past solutions and search-based feedback [87-92]. The agent was designed to identify unnecessary code found in variables, functions, and classes based on automated tools like UCDetector [93]. If the participant asked for the location of variables/classes/methods within the code, the agent's ability to respond was based on both static and dynamic feature location techniques [94, 95]. Generating test cases automatically (i.e., without past solutions) was done by either code search algorithms or converting user stories to scenarios and then to test cases [96-99]. The agent could identify missing code using a technique investigated in the Haskell programming language [100].

Creative thinking is essential to a programmer's success especially when solving open-ended problems [18, 101-108]. To encourage diversity in thinking the agent offered abstract code templates, code examples, and alternate implementations. This decision was motivated by Tsuei's research that imperfect guidance enhances creativity and encourages exploration of new ideas [109]. Creativity theory suggests the production of a large number of ideas to arrive at creative ones through ideational fluency, since agents themselves cannot ideate [110]. Thus, the agent may ask things like, "*Are there other ways to do this?*"

The agent's responses were tuned to its performance with the partner giving just-in-time help when needed; it also apologized for incorrect or unknown answers, expressed uncertainty, and gave who/what/when/why/where/how answers accompanied with directions or suggestions [111, 112]. The agent also provided verbal feedback by presenting content and code templates to help the programmer's memory. For example, the agent corrected a participant by highlighting code and giving verbal suggestions.

Doing so addressed a self-presentation bias that induces a lack of memory retention on the human's part [113 - 116].

Study Design

14 participants (7 men and 7 women) were recruited for the study through advertisements and a recruitment site called Upwork. Eight of our participants were university students (4 men and 4 women) and six were professional programmers (3 men and 3 women). This study was conducted during the COVID-19 pandemic. Therefore, virtual lab studies were conducted from participants throughout United States.

The study design was like the human-human study, except (1) the participant completed the task with an agent, (2) they were given an instructional tutorial on the agent, (3) no pair jelling was incorporated, and (4) the agent changed gender presentation halfway through the study. We split the gender embodiment of the agent equally among the study sessions to counterbalance. Each participant signed a consent form at the end of the study, as it was a deception study.

2.3 Analysis of Data

Audio and Video of each session was recorded in both human-human and human-agent studies. These recording were then transcribed into individual snippets of dialogue which were subsequently sorted by their intent. The intent of each was represented by dialogue acts as described in Table 1. "Dialogue acts" have been used to classify human utterances using criteria based on a combination of pragmatics, semantics, and syntaxes [47]. Dialogue acts enable us to understand verbal communication as well as conversations via Hidden Markov Models [47], which are necessary for a conversational agent. 20% of the transcripts were coded by three researchers. Once they reached the inter-rater reliability of 90% (using the Jaccard measure [48]), two researchers independently coded the rest of the transcripts. We analyzed the data qualitatively and quantitatively to understand the differences of communication styles between humans with humans and humans with an agent.

Table 1. The dialogue acts and their definitions as used to code the study transcriptions.

Dialogue Acts	Definition
Abandoned	An Unfinished Remark
Acknowledgement	Acceptance of the existence of something
Answer No	"No" responses
Apology	A regretful acknowledgement of failure
Answer W/H Questions	Answering who/what/when/where/why/how questions
Answer Yes	"Yes" Responses
Direct Instruction	An explicit instruction
Feedback Non-Positive	A non-positive response or comment

Feedback Positive	A positive response or comment
Indirect Instruction	Implicit or polite instruction
Other (Filler Words)	Meaningless words
W/H/ QUESTION	A who/what/when/where/why/how question
Questions Yes/No	Questions asking for a yes/no answer
Statement	A declaration or remark
Uncertainty	Dialogue that indicates uncertainty

3 Results

The results of the studies: both similarities and differences in the communication styles (dialogue acts) are summarized in Table 2. The differences between the studies in regard to the research questions are discussed subsequently.

Table 2. The dialogue acts for human-human and human-agent studies. The most prominent difference in dialogue acts is highlighted. Yellow for humans with humans vs. agent (RQ1), peach for women with human vs. agent (RQ2), and blue for men with human vs. agent (RQ3).

Dialogue Acts	**Human-Human**				**Human-Agent**			
	M[#]	**W[#]**	**Total**	**%age**	**M[#]**	**W[#]**	**Total**	**%age**
Abandoned	97[9]	108[8]	205	6.01	104[7]	101[7]	205	6.56
Acknowledgement	295[9]	326[9]	621	18.22	129[7]	166[7]	295	9.43
Answer No	5[6]	16[6]	21	0.62	6[7]	3[3]	9	0.29
Apology	7[6]	10[5]	17	0.50	5[5]	4[4]	9	0.29
Answer W/H Questions	21[9]	19[5]	40	1.17	65[7]	62[7]	127	4.06
Answer Yes	52[9]	51[9]	103	3.02	70[7]	55[7]	125	4.00
Direct Instruction	87[9]	72[9]	159	4.66	46[7]	81[7]	127	4.06
Feedback Non-Positive	23[8]	14[5]	37	1.09	7[5]	4[4]	11	0.35
Feedback Positive	58[9]	26[7]	84	2.47	23[6]	49[7]	72	2.30
Indirect Instruction	129[9]	89[9]	218	6.39	78[7]	120[7]	198	6.33
Other (Filler Words)	126[9]	112[9]	238	6.98	46[7]	61[7]	107	3.42
W/H Questions	48[9]	62[8]	110	3.23	82[7]	96[7]	178	5.69
Questions Yes/No	97[9]	119[9]	216	6.34	84[7]	125[7]	209	6.68
Statement	688[9]	592[9]	1280	37.55	657[7]	753[7]	1410	45.09
Uncertainty	27[9]	33[9]	60	1.76	16[7]	29[5]	45	1.44
Total	**1760**	**1649**	**3409**		**1418**	**1709**	**3127**	

RQ1: What communication styles are used by programmers when interacting with a human vs. an agent?
To answer RQ1, we compared the dialogue acts used by participants (both men and women) with another participant and with our agent. Table 2 shows (in yellow) the

frequencies of different dialogue acts for human-human vs. human-agent. The different communication styles found between human-human vs. human-agent were:

More filler words with humans vs. fully articulated thoughts with an agent.

Participants in the human-human study used 55% more filler words (see Table 2) to give their verifications or to fill the gaps in the communication. For example, when HH1-M2 was implementing the horizontal win condition, M1 responded with *"Um" "Ok,"* or *"Huh"* while responding to M2's thought process. Conversely, while communicating with agent partners, participants fully formulated their thoughts before articulating them. The agent themselves never used filler words, as they followed a script.

Non-positive feedback and acknowledgment to other humans vs. agents

In the human-human study, participants used non-positive feedback 70.3% more than their counterparts in the human-agent study. For example, HH9-M8 gave non-positive feedback for implementation of the win method with a sarcastic comment: *"okay, so it's a failure, awesome."* Similarly, HH7-W8 gave non-positive feedback to her partner's idea when she stated "*Yeah, I'm not sure about that one."* The decreased usage of non-positive feedback in the human-agent study stemmed from our design choice of agent being motivational, empathetic, and a rapport-builder.

The human-human study participants acknowledged their partners' ideas 52.5% more than human-agent participants. For example, in HH9, M8 shared a possible solution for checking diagonal win conditions and suggested the usage of multiple 'loops', to which his partner M9 responded "*Yeah we could do something like that.*" Later in the same study, they switched roles, describing a test method he (M9) might want to implement and M8 confirming "*Yeah, I think that'll work*." These types of dialogue acts did not appear as frequently in the human-agent studies. One possible reason for this could be that when conversing with another person acknowledgement is key to effectively communicating ideas [49] and hence, in the absence of another human the acknowledgement was decreased.

Asked more WH Questions from the agent vs. human

Participants from the human-agent study also asked 38.2% more W/H (who/what/when/why/where/how) questions to an agent as compared to the human-human study. For example, after getting stuck, HA-M6 prompts the agent by commenting, "*What do you think?*" and later with "*How about you drive*." Additionally, HA-W13 asked questions such as "*Why would you return true?*" "*How do we check*?" The WH question were asked by participants to clarify, understand the code, prompt for feedback, or ask for help.

Human participants were more reliant on the agent as they had more confidence in the agent's responses and solutions. For example, HA-M3 asked the agent, "*What do you think the mistake is?*" The agent replied with, "*You have a missing bracket on line 66*". This helped participants quickly find out the error and save time. HA-M3 also asked, "*What's the next story?*" or "*What's story number four?*" This helped the

participants navigate quickly between tasks. Participants in human-agent study were more comfortable with the agent and saw it as a non-judgmental partner [4].

RQ2: **How do men's communication styles differ when interacting with a human vs. an agent?**

We found the following differences (highlighted in peach in Table 2) in men's communication style with another human (man/woman) vs. an agent.

Men showed more uncertainty with another human vs. an agent

Men participants in the Human-Human study were 40.7% more uncertain about the task. For example, in HH2, both participants were uncertain about the next step, as M3 commented "*Ya I don't know*", to which M4 responded "*I'm not sure how we can write a test for this because it's not going to evaluate to true or false you know*." Humans trust agents more than other humans [50], and this was evident as HA-M8 commented "*the computer knows more … than a human knows… I would trust a computer more*."

Men gave more positive feedback to their human partners

Men participants of the human-human study showed 60.3% more positive feedback, listened to their partners, and acknowledged their suggestions. All nine men participants showed positive feedback during the human-human study. For example, in HH1, while working on method to place marks on a board, M1 laughed and commented "*It's ugly but it should work*" to which M2 responded with a laugh and commented "*Yeah, that's how I feel sometimes"*. The frequency of acknowledgement was twice as high in the human-human study than in the human-agent study. For example, in HH2, M4 was describing how to keep track of current player, placing the mark. M3 listened carefully and acknowledged HH2-M4's thought process without abandoning it mid-point.

Men gave more direct and indirect instructions to another human partner than an agent

Men participants gave 51.8% more direct and indirect instructions to another human partner than to an agent. Direct instructions are an explicit way of expressing what needs to be done while indirect instructions are a subtle way of suggesting things that need to be done. Direct instructions were used to express what aspect of the problem needs to be done and how to accomplish it. For example, in HH3, when W1 tried to explain the logic for checking if marks are in the same row verbally, M5 explicitly said "*Uh, write it out for me. I, I can visualize it a bit better if I can see it,"* Indirect instructions were used for directions, suggestions, agreeing to tasks, and giving control. HH1-M1 gave indirect instructions to M2 and commented "*We could probably use the code for tie winner, but that's my guess.*"

RQ3: How do women's communication styles differ when interacting with a human vs. an agent?

We found the following differences (highlighted blue in Table 2) in women's communication style with another human (man/woman) vs. an agent. Women participants talked more with an agent, as the number of statements was 21.4% more than with another human.

Women used more direct and indirect instructions with an agent than with another human

Women participants used 19.9% more instructions (both direct and indirect) to direct the flow of the task process with an agent. For example, HA-W4 was more vocal and direct with the agent. She was controlling the flow of dialogue with "*Do it again*", "*Sure, let's do that*", and "*Try it*". She also used indirect instructions to implicitly direct the flow as she commented "*Okay, Um, so we need to add test*", "*I think there needs to be more code in check ties*", and "*Should we just try placing*".

Women apologized more with a human partner than with an agent

Women participants tended to apologize with a human partner when an answer was unknown or wrong. While working with an agent, women were less self-conscious when they made mistakes and were more confident in their approaches. For example, in HH4, W3 apologized to her partner for simple things like clicking by mistake as she commented "*Sorry, I'm clicking*". Likewise, HH4-W2 read the task wrong and she apologized immediately "*Oh, sorry,…oh my gosh. I'm so sorry. I read that wrong*." Women build rapport with their human partner, not wanting to let their partner down [52, 53].

Women disagreed with another human with a no

Women participants disagreed (with a "No") 81% more with a human partner than an agent. For example, in HH4, W2 asked if they need to write a test case for mark placed, to which her partner W3 responded "*No no no...*"

Women positively responded to an agent

Women participants gave 46.9% more positive feedback to an agent than to a human partner. For example, HA-W4 responded positively to agent's code by commenting, "*It does make more sense*."

4 Discussions

Our results shed light on the conversational styles of humans, both men and women, with other humans and agents. As seen in RQ2 and RQ3 the communication styles were different for both men and women, hence, to create a gender-inclusive programming agent we should integrate styles from both genders. Some of the implications for programming conversational agents are detailed as follows.

Conversational agents for problem solving tasks should support WH questions and answers, WH answers should be accompanied by directions and suggestions. In our study, while answering, humans tended to answer yes/no more frequently but the agent should explain the logic behind the code or explanations regarding the decisions made

by the agent. HA-M6 mentioned that the agent should give a "*more specific, explanatory approach like a human*." Though generating human-like explanation and discussions regarding programming may be hard to implement with the current technology, utilizing the static and dynamic testing approaches, generating visually explainable solutions can make the agent more human-like. Further, such plausible explanations can help to build trust of both professionals and students with an agent and in its solutions [50], while a lack of such explanations may affect losing trust. The agent should show vulnerabilities of being a machine and expose its limitations to increase trust with a human partner. When a human shows uncertainty, the agent should better explain using visualizations and the underlying concepts of how it arrived at a specific solution. Additionally, the agent should provide better verbal feedback and present content and code templates to jog the programmer's memory. For example, to correct a programmer, the agent should highlight the code segments while giving verbal and visual suggestions about the code. One approach to generate such explanations could be using deep learning techniques and training them on WH questions. However, such training will require tremendous amount of data on pair programming conversations that currently is unavailable. Further, supporting gender specific problem solving [37-39], and leadership styles [2] should also be facilitated by a gender-inclusive agent.

Engaging the human partner with both positive and negative feedback was important for the success of the task. The feedback from human as well as agent was regarding (1) status of the code i.e., pass/fail/unexpected results, (2) code reviews i.e., correct/incorrect/coding style, (3) idea implemented work/fails, (4) agreement, and (5) motivation. The positive feedback helped human partners to stay motivated and engaged in the task, while negative feedback was on mistakes in the code and helped them to improve quality of the code. Both positive and negative feedback are important towards problem solving tasks and hence should be used as integral parts of the conversational agents. Since there were differences in how each gender used feedback, it will be important that a gender-inclusive agent is able to adapt itself based on past conversations. This is especially important regarding negative feedback, which should be dealt more carefully. For-example, if there is a typo, the agent can correct the typo by itself but if there is a logical error or the problem-solving approach is wrong, the agent should give suggestions.

One implication for machine learning algorithms is to train them on different genders (equal number of men, women and non-binary) to capture the diversity of conversational styles. The differences between the communication styles between different genders necessitate inclusion of features in the machine learning algorithms that can capture these differences. Hence, avoiding agents that support misogynistic ideologies in machine learning [33-36] and support more gender-inclusive machine learning algorithms.

5 Conclusion

We take the first step towards creating a gender-inclusive Alexa-like programming partner. This paper contributes to understanding the differences between human-human and human-agent conversations, and how gender effects conversations.

1. **RQ1: Communication Styles: Human vs. Agent.** Participants in human-human study gave more directions to complete the task and said more filler words. They also expressed non-positive feedback and acknowledged their partners' ideas and interjected when they thought they could help. They also asked more W/H questions as they were comfortable with the agent.
2. **RQ2: Men's Communication Styles: Human vs. Agent.** Men participants motivated their human partner through positive feedback and giving direction using both direct and indirect instructions. Men participants also showed more trust with the agent.
3. **RQ3: Women's Communication Styles: Human vs. Agent.** Women participants disagreed and apologized more with a human partner while making rapport with them. They instructed the agent and gave more positive feedback. Hence, women participants were hesitant to communicate with human partners.

Finally, we discuss the implications for a gender-inclusive conversational agent. The implications include interface design of such an agent as well as for training machine-learning algorithms.

References

1. P. Robe, S. K. Kuttal, Y. Zhang, R. Bellamy.: Can Machine Learning Facilitate Remote Pair Programming? Challenges, Insights & Implications. In: Proceedings of Visual Languages and Human-Centric Computing, (2020).
2. S. K. Kuttal, K. Gerstner, A. Bejarano.: Remote Pair Programming in Online CS Education.: Investigating through a Gender Lens. In: Proceedings of Visual Languages and Human-Centric Computing, (2019).
3. S. K. Kuttal, J. Myers, S. Gurka, D. Magar, D. Piorkowski, R. Bellamy.: Towards Designing Conversational Agents for Pair Programming: Accounting for Creativity Strategies and Conversational Styles. In: Proceedings of Visual Languages and Human-Centric Computing, (2020).
4. S. K. Kuttal, K. Kwasny, B. Ong, P. Robe.: Understand the tradeoffs for substituting humans with an agent - good, bad, and ugly. Submitted to CHI 2021 found at https.://drive.google.com/drive/folders/14_0zkttwbVr6pJnB_U4YIReGDLI6mCTX?usp=sharing
5. P. Robe, S. K. Kuttal.: Designing an interactive pair programming partner submitted to TOCHI 2021 found at https.://drive.google.com/drive/folders/1vIOdro0pg8C1jSB42KzYrDRKO0PVhqZ1?usp=sharing
6. Andreas Stolcke, Noah Coccaro, Rebecca Bates, Paul Taylor, Carol Van Ess-Dykema, Klaus Ries, Elizabeth Shriberg, Daniel Jurafsky, Rachel Martin, Marie Meteer.: Dialogue act

modeling for automatic tagging and recognition of conversational speech. Comput. Linguist. 26, 3, 339-373. (2000)
7. Alina Tugend.: Why is asking for help so difficult? The New York Times (2007).
8. PairBuddy Github, https.://github.com/grubtub19/pairbuddy
9. A. Abraham.: Gender and creativity.: An Overview of Psychological and Neuroscientific Literature. Brain Imaging and Behavior 10(2), 609-618, (2016).
10. S. Baron-Cohen, R. C. Knickmeyer, M. K. Belmonte.: Sex differences in the brain: implications for explaining autism. Science, 310(5749), 819–823 (2005).
11. A. LeClair, Z. Eberhart, C. McMillan.: Adapting Neural Text Classification for Improved Software Categorization. IEEE International Conference on Software Maintenance and Evolution (ICSME), Madrid, 461-472. (2018)
12. W.-L. Lin, K.-Y. Hsu, H.-C. Chen, J.-W. Wang.: The relations of gender and personality traits on different creativities.: a dual-process theory account. Psychology of Aesthetics, Creativity, and the Arts, 6(2), 112–123 (2012).
13. A. Wood, P. Rodeghero, A. Armaly, C. McMillan.: Detecting speech act types in developer question/answer conversations during bug repair. In Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2018), 491-502 (2018).
14. W. Woolley, I. Aggarwal, T. W. Malone.: Collective intelligence and group performance. Current Directions in Psychological Science, 24.6, pp.420-424 (2015).
15. D. W. Palmieri.: Knowledge management through pair programming, Master's Thesis, Department of Computer Science, North Carolina State University, Raleigh, NC, (2002).
16. L. Williams, C. McDowell, N. Nagappan, J. Fernald, L. Werner.: Building pair programming knowledge through a family of experiments. In: 2003 International Symposium on Empirical Software Engineering, pp. 143–152 (2003).
17. L. Williams, R. Kessler.: Pair Programming Illuminated. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, (2002).
18. C. Barra, B. Crawford.: Fostering creativity thinking in agile software development, Vol. 4799, pp. 415–426 (2007).
19. A. Belshee.: Promiscuous pairing and beginner's mind: Embrace inexperience, pp. 125 – 131 (2005).
20. A. Cockburn, L. Williams.: Extreme programming examined. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, Ch. The Costs and Benefits of Pair Programming, pp. 223–243 (2001).
21. T. DeMarco, T. Lister.: Peopleware.: Productive Projects and Teams. Dorset House Publishing Co., Inc., New York, NY, USA (1987).
22. F. Zieris, L. Prechelt.: On knowledge transfer skill in pair programming. In: Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '14, ACM, New York, NY, USA, 2014, pp. 11:1–11:10.
23. C. McDowell, L. Werner, H. Bullock, J. Fernald.: The effects of pair-programming on performance in an introductory programming course. In: Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education, pp. 38–42. SIGCSE, ACM, New York, NY, USA (2002).
24. N. Katira, L. Williams, L. Williams, E. Wiebe, C. Miller, S. Balik, E. Gehringer.: On understanding compatibility of student pair programmers, SIGCSE Bull. 36 (1), pp. 7–11 (2004).
25. C. McDowell, L. Werner, H. E. Bullock, J. Fernald.: The impact of pair programming on student performance, perception and persistence. In: Proceedings of the 25th International

Conference on Software Engineering, ICSE '03, IEEE Computer Society, Washington, DC, USA, pp. 602–607 (2003).
26. L. Williams, E. Wiebe, K. Yang, M. Ferzli, C. Miller.: In support of pair programming in the introductory computer science course. Computer Science Education, 12, pp. 197–212 (2002).
27. O. Ruvalcaba, L. Werner, J. Denner.: Observations of pair programming: Variations in collaboration across demographic groups. In: Proceedings of the 47th ACM Technical Symposium on Computing Science Education, SIGCSE, ACM, New York, NY, USA, pp. 90–95 (2016).
28. L. L. Werner, B. Hanks, C. McDowell.: Pair-programming helps female computer science students. J. Educ. Resour. Comput. 4 (1), (2004).
29. M. Celepkolu, K. E. Boyer.: Thematic analysis of students' reflections on pair programming in cs1. In: Proceedings of the 49th ACM Technical Symposium on Computer Science Education, pp. 771–776, SIGCSE, ACM, New York, NY, USA (2018).
30. F. J. Rodríguez, K. M. Price, K. E. Boyer.: Exploring the pair programming process: Characteristics of effective collaboration. In: Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, pp. 507–512, SIGCSE '17, ACM, New York, NY, USA, (2017).
31. Judith B.: Revisiting Bodies and Pleasures: Theory, Culture & Society 16, 2, pp. 11–20 (1999).
32. Candace W. and Don H. Z.: Doing Gender. Gender & Society 1, 2, pp. 125–151 (1987).
33. Margaret B., Anicia P., Charles H., and Noha E.: Finding Gender-Inclusiveness Software Issues with GenderMag: A Field Investigation. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems. ACM, pp. 2586–2598 (2016)
34. Gary C., Uri G.: Strong Evidence for Gender Differences in Risk Taking. Journal of Economic Behavior & Organization 83, 1, pp. 50–58 (2012),
35. Christopher M., Hema S. P., Zoe S.-H., Claudia H., Amber H., Charles H., Logan S., Nupoor P., Anita S., and Margaret B.: Open-Source Barriers to Entry, Revisited: A Sociotechnical Perspective. In: Proceedings of the 40th International Conference on Software Engineering, ACM, pp. 1004–1015 (2018)
36. Arun S., Nicola M.: Cognitive Walkthrough of a Learning Management System with Gendered Personas. In: Proceedings of the 4th Conference on Gender & IT. ACM, pp. 191–198 (2018).
37. Susan L.: 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, pp. 14–16 (Gothenburg, Sweden) (GE '18). Association for Computing Machinery, NewYork, NY, USA,. (2018)
38. E. Arisholm, H. Gallis, T. Dybå, and D. I. K. Sjoberg.: Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise. IEEE Transactions on Software Engineering 33, 2, pp. 65–86. (2007)
39. Katrina F., Nickolas F., Rebecca V.: Collaborative learning and anxiety: a phenomenographic study of collaborative learning activities. In: Proceedings of the 44th ACM Technical Symposium on Computer Science Education, pp. 227–232 (2013).
40. Virtual Assistant [n.d.]. Amazon Alexa. https.://developer.amazon.com/en-US/alexa
41. Virtual Assistant [n.d.]. Apple Siri. https.://www.apple.com/siri/
42. Virtual Assistant [n.d.]. Google Assistant. https.://assistant.google.com/
43. Social Bot [n.d.]. Cleverbot. https.://www.cleverbot.com/
44. Social Bot [n.d.]. Mitsuku. https.://www.pandorabots.com/mitsuku/

45. Social Bot [n.d.]. SAP Conversational AI. https.://www.sap.com/products/conversational-ai.html
46. Social Bot [n.d.]. Xiaoice AI Assistant. https.://www.digitaltrends.com/cool-tech/xiaoice-microsoft-future-of-ai-assistants/
47. Andreas S., Noah C., Rebecca B., Paul T., Carol V. E.-D., Klaus R., Elizabeth S., Daniel J., Rachel M., and Marie M.: Dialogue act modeling for automatic tagging and recognition of conversational speech. Comput. Linguist. 26, 3, pp. 339–373 (2000).
48. Jaccard P.: Étude comparative de la distribution florale dans une portion des Alpes et des Jura. Bulletin de la Société vaudoise des sciences naturelles. 37, pp. 547–579 (1901).
49. Kathryn R. W., Deborah E. W.: Peer Relationships and Collaborative Learning as Contexts for Academic Enablers. School Psychology Review, 31, 3, pp. 366-377 (2002).
50. S.T. Fiske, E.H.P.P.S.T. Fiske, and S.E. Taylor.: Social Cognition. McGraw-Hill, New York City, USA.
51. S. K. Kuttal, B. Ong, K. Kwasny, P. Robe.: Trade-offs for Substituting a Human with an Agent in a Pair Programming Context: The Good, the Bad, and the Ugly. In: Proceedings of the conference on Human Factors in Computing, CHI (2021).
52. I. Cuadrado, M. M. D. Navas, F. Molero, E. Ferrer, J. F. Morales.: Gender differences in leadership styles as a function of leader and subordinates sex and type of organization, 2012.
53. T. Yang, H. E. Aldrich.: Whos the boss? explaining gender inequality in entrepreneurial teams, American Sociological Review, 79 (2), pp. 303–327 (2014).
54. Prashant Baheti, Dr Gehringer, and P. Stott.: Exploring the Efficacy of Distributed Pair Programming. (2002).
55. Rafael D., Crescencio B.: Analyzing Work Productivity and Program Quality in Collaborative Programming. In: Proceedings of the 2008 The Third International Conference on Software Engineering Advances, pp. 270–276, IEEE Computer Society, Washington, DC, USA (2008).
56. Brian H.: Empirical evaluation of distributed pair programming. International Journal of Human-Computer Studies, 66, pp. 530–544 (2008).
57. Deborah R. C., Christopher A. H.: Computer Self-Efficacy: Development of a Measure and Initial Test. MIS Q.19, 2, pp. 189–211. (1995)
58. Clayton L.: Using the "thinking-aloud" method in cognitive interface design. IBM T.J. Watson Research Center, Yorktown Heights, N.Y (1982).
59. Danielle L. J., Scott D. F.: What use is a backseat driver? A qualitative investigation of pair programming. In: Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing, pp. 103–110, VL/HCC, (2013).
60. Morae 2019. Morae. http.://www.techsmith.com/morae.asp
61. Timothy B., Justine C.: 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), 396–403. ACM, New York, NY, USA (2001).
62. Jay B., David B., Oli M., Nick W.: Wizard of Oz experiments and companion dialogues. In: Proceedings of the 24th BCS Interaction Specialist Group Conference, pp. 117–123. British Computer Society, (2010).
63. Nils D., Arne J., and Lars A.: Wizard of Oz studies—why and how. Knowledge-based systems 6, 4, pp. 258–266 (1993).
64. Pierre W., Giovanni C., Yann L.-C., Samuel B., Pierre J., Anne-Sophie R.: 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, pp. 1–8. (SeGAH). IEEE, (2016).
65. Software Application [n.d.]. Facerig. Https.://facerig.com/

66. Laurel D. R.: Wizard of oz studies in hri: a systematic review and new reporting guidelines. Journal of Human-Robot Interaction 1, 1, 119–136 (2012).
67. Zahra A., Mohit J., Q. Vera Liao, Justin D. W.: Resilient Chatbots: Repair Strategy Preferences for Conversational Breakdowns. In: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, Article 254, 12 pages. (Glasgow, Scotland Uk)(CHI '19). Association for Computing Machinery, New York, NY, USA (2019).
68. Irene L., Harriet W.: Personification of the Amazon Alexa: BFF or a Mindless Companion. In: Proceedings of the 2018 Conference on Human Information Interaction & Retrieval, pp. 265–268. (New Brunswick, NJ, USA)(CHIIR '18). Association for Computing Machinery, New York, NY, USA (2018).
69. Lee S., Mani S., Sara K., Janet H. W., Keith W.: When the Interface is a Face. Hum.-Comput. Interact.11, 2, 97–124. (1996).
70. Mohan Z., Julia W., Amanpreet K., Benjamin L.: Assessing the Impact of Virtual Human's Appearance on Users' Trust Levels. In: Proceedings of the 18th International Conference on Intelligent Virtual Agents, 329–330. (Sydney, NSW, Australia)(IVA '18). Association for Computing Machinery, New York, NY, USA (2018).
71. Jonathan G., Ning W., Jillian G., Edward F., Robin D.: 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, pp. 125–138. Berlin, Heidelberg, (2007).
72. Dai H., Justine C, Kenji A.: The Role of Embodiment and Perspective in Direction-Giving Systems. (2010).
73. Ameneh S., Q. Vera Liao, Dakuo W., Rachel K. E. B., Thomas E.: 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, pp. 1–13. (Montreal QC, Canada)(CHI '18).Association for Computing Machinery, New York, NY, USA (2018).
74. Akikazu T., Taketo N.: Situated Facial Displays: Towards Social Interaction. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systemsm, pp. 450–455. (Denver, Colorado, USA)(CHI '95). ACM Press/Addison-Wesley Publishing Co., New York, NY, USA (1995).
75. Susanne v. M., Elisabeth A., and Jochen M.: The Persona Effect: How Substantial Is It?. In People and Computers XIII, Hilary Johnson, Lawrence Nigay, and Christopher Roast (Eds.) pp. 53–66. Springer London, London (1998).
76. Nick Y., Jeremy N B., Kathryn R.: 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, pp. 1–10. (San Jose, California, USA)(CHI'07). ACM, New York, NY, USA (2007).
77. Saros [n.d.]. Saros Project. https.://www.saros-project.org/
78. Peter H. K., Nathan G. F., Takayuki K., Hiroshi I., Jolina H. R., Rachel L. S., Shaun K. K.: Design Patterns for Sociality in Human-Robot Interaction. In: Proceedings of the 3rd ACM/IEEE International Conference on Human Robot Interaction, 97–104. (Amsterdam, The Netherlands) (HRI '08). Association for Computing Machinery, New York, NY, USA (2008).
79. Mohit J., Pratyush K., Ishita B., Q. Vera Liao, Khai T., and Shwetak P.: Farm Chat: A Conversational Agent to Answer Farmer Queries. Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 2, 4, Article 170, 22 pages (2018).
80. Mohit J., Pratyush K., Ramachandra K., Shwetak N P.: Evaluating and informing the design of chatbots. In: Proceedings of the2018 Designing Interactive Systems Conference, pp. 895–906. (2018).

81. Ewa L. Abigail S.: “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, pp. 5286–5297. (2016).
82. Teresa M. A., Michael G. P.: The dynamic componential model of creativity and innovation in organizations: Making progress, making meaning. Research in Organizational Behavior 36, 157 – 183 (2016).
83. Michael A.: Armstrong’s handbook of reward management practice: Improving performance through reward (12 ed.). Kogan Page Publishers, (2012).
84. Christopher P C., Jessica M N., Michael T F.: Intrinsic motivation and extrinsic incentives jointly predict performance: A 40-year meta-analysis. Psychological bulletin 140, 4, 980 (2014).
85. Edward L D., Anja H O., Richard M R.: Self-determination theory in work organizations: The state of a science. Annual Review of Organizational Psychology and Organizational Behavior 4, 19–43 (2017).
86. Carmen F., Charlotte P. M., Ernestine S.: The Influence of Intrinsic Motivation and Synergistic Extrinsic Motivators on Creativity and Innovation. Frontiers in Psychology 10, 137 (2019)
87. M. Day, M. R. Penumala, and J. Gonzalez-Sanchez: Annete: An Intelligent Tutoring Companion Embedded into the Eclipse IDE. In: 2019 IEEE First International Conference on Cognitive Machine Intelligence, 71–80. (CogMI). (2019).
88. Iman K., Juergen R., Ying Z.: Spotting Working Code Examples. In: Proceedings of the 36th International Conference on Software Engineering, 664–675. Hyderabad, India, Association for Computing Machinery, New York, NY, USA (2014).
89. Kisub K., Dongsun K., Tegawendé F. B., Eunjong C., Li L., Jacques K., Yves Le T.: FaCoY: A Code-to-Code Search Engine. In: Proceedings of the 40th International Conference on Software Engineering, pp. 946–957. (Gothenburg, Sweden) (ICSE ’18). Association for Computing Machinery, New York, NY, USA (2018).
90. Haoran N., Iman K., Ying Z.: Learning to rank code examples for code search engines. Empirical Software Engineering 22, 1, 259–291 (2017).
91. M. Raghothaman, Y. Wei, and Y. Hamadi: SWIM: Synthesizing What I Mean - Code Search and Idiomatic Snippet Synthesis. In: 2016 IEEE/ACM38th International Conference on Software Engineering, pp. 357–367. ICSE (2016).
92. R. Zhi, S. Marwan, Y. Dong, N. Lytle, T. W. Price, T. Barnes: Toward Data-Driven Example Feedback for Novice Programming. In: Proceedings of the 12th International Conference on Educational Data Mining. (2019).
93. Jörg Spieler. [n.d.].UCDetector. http.://www.ucdetector.org/
94. Dapeng L., Andrian M., Denys P., Vaclav R.: Feature location via information retrieval based filtering of a single scenario execution trace. In: Proceedings of ASE’07 - 2007 ACM/IEEE International Conference on Automated Software Engineering, 234–243. (2007).
95. T. Savage, M. Revelle, and D. Poshyvanyk: FLAT3: feature location and textual tracing tool. In: Proceedings of 2010 ACM/IEEE 32nd International Conference on Software Engineering, Vol. 2. pp. 255–258. (2010).
96. S. Ali, L. C. Briand, H. Hemmati, R. K. Panesar-Walawege: A Systematic Review of the Application and Empirical Investigation of Search-Based Test Case Generation. IEEE Transactions on Software Engineering 36, 6, 742–762 (2010).
97. Meiliana, Irwandhi Septian, Ricky Setiawan Alianto, Daniel, Ford Lumban Gaol.: Automated Test Case Generation from UML Activity Diagram and Sequence Diagram using Depth First Search Algorithm. Procedia Computer Science 116, pp. 629 – 637 (2017).

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).
98. Matheus M., Erica S., Andre E., Nandamudi V.: Analyzing graph-based algorithms employed to generate testcases from finite state machines. (2019).
99. P. Rane: Automatic Generation of Test Cases for Agile using Natural Language Processing. (2017).
100. Alex G., Bastiaan H., Johan J., L. Thomas v. B.: Ask-Elle: an Adaptable Programming Tutor for Haskell Giving Automated Feedback. International Journal of Artificial Intelligence in Education. 27, (2016).
101. Tim B.: Change by Design: How Design Thinking Transforms Organizations and Inspires Innovation. Harper Business. (2009).
102. Berland Edelman and Inc. 2010. Creativity and education.: Why it matters http.://www.adobe.com/aboutadobe/pressroom/pdfs/Adobe_Creativity_and_Education_Why_It_Matters_study.pdf, last accessed 2019/09/18
103. Marvin L.: Effective problem solving. Prentice Hall. (1988).
104. Zhiqiang L., Dieter J S.: Teaching creativity in engineering. International Journal of Engineering Education 20, 5, 801–808 (2004).
105. George Polya.: How to solve it.: A new aspect of mathematical method. Vol. 85. Princeton university press. (2004).
106. Tony W., Robert A C.: Creating innovators: The making of young people who will change the world. Simon and Schuster. (2012).
107. Wayne A W.: How to solve problems: Elements of a theory of problems and problem solving. WH Freeman San Francisco. (1974).
108. Yong Z.: World class learners: Educating creative and entrepreneurial students. Corwin Press, Thousand Oaks, California, USA (2012).
109. Mengping T.: 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, 147–161 (2017)
110. J.P. Guilford.: Intelligence, Creativity, and Their Educational Implications. R. R. Knapp. (1968) https.://books.google.com/books?id=WE8kAQAAMAAJ
111. T. Robertson, Shrinu P., Margaret B., Curtis C., Joe R., Laura B., Amit P.: Impact of Interruption Style on End-User Debugging. ACM Conference on Human Factors in Computing Systems, 287–294 (2004).
112. Aaron W., Margaret B., Laura B., Orion G., Ledah C., Curtis C., Mike D., Gregg R.: Harnessing Curiosity to Increase Correctness in End-User Programming. 305–312 (2003)
113. Dominique K. Ludovic L. B.: The Influence of Reference Acceptance and Reuse on Conversational Memory Traces. Journal of experimental psychology. Learning, memory, and cognition 4, (2014).
114. Dominique K., Ludovic L. B., Christine R.: Explicit feedback from users attenuates memory biases in human-system dialogue. International Journal of Human-Computer Studies 97, 77 – 87 (2017). http.://www.sciencedirect.com/science/article/pii/S1071581916301045
115. Dominique K., Christine R., Ludovic L. B.: Generating References in Naturalistic Face-to-Face and Phone-Mediated Dialog Settings. Topics in Cognitive Science, 8, (2016).
116. R. Sharma, S. Gulia, and K. K. Biswas: Automated generation of activity and sequence diagrams from natural language requirements. In: 2014 9th International Conference on Evaluation of Novel Approaches to Software Engineering, 1–9. ENASE, (2014).
117. TeamViewer 20219, Teamviewer. https.://www.teamviewer.com/
118. Eclipse 2019, Eclipse Foundation https.://www.eclipse.org/ide