Towards Designing Conversational Agents for Pair Programming: Accounting for Creativity Strategies and Conversational Styles

Sandeep Kaur Kuttal
University of Tulsa
Tulsa, OK, United States
sandeep-kuttal@utulsa.edu

Jarow Myers
University of Tulsa
Tulsa, OK, United States
jgm5877@utulsa.edu

Sam Gurka
University of Tulsa
Tulsa, OK, United States
sag7778@utulsa.edu

David Magar
University of Tulsa
Tulsa, OK, United States
ddm2084@utulsa.edu

David Piorkowski
IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
dpiorkowski@gmail.com

Rachel Bellamy
IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
rachel@us.ibm.com

***Abstract*—Established research on pair programming reveals benefits, including increasing communication, creativity, self-efficacy, and promoting gender inclusivity. However, research has reported limitations such as finding a compatible partner, scheduling sessions between partners, and resistance to pairing. Further, pairings can be affected by predispositions to negative stereotypes. These problems can be addressed by replacing one human member of the pair with a conversational agent. To investigate the design space of such a conversational agent, we conducted a controlled remote pair programming study. Our analysis found various creative problem-solving strategies and differences in conversational styles. We further analyzed the transferable strategies from human-human collaboration to human-agent collaboration by conducting a Wizard of Oz study. The findings from the two studies helped us gain insights regarding design of a programmer conversational agent. We make recommendations for researchers and practitioners for designing pair programming conversational agent tools.**

I. Introduction

Pair programming exercises the principle that "two heads are better than one" [1]. In pair programming, two programmers solve a problem together co-located (sitting shoulder-to-shoulder at a computer) or remote (sharing the computer screen and communicating remotely). One partner acts as the *driver*, who actively creates code and controls the keyboard and mouse; whereas, the other partner is the *navigator*, who constantly reviews the driver's work, proposes suggestions, and asks clarifying questions [2]–[4]. There is a long history of established research on pair programming's benefits such as increased communication, creativity, knowledge management, and self-efficacy [2]–[14]. Pair programming reduces the gender gap as it encourages women to pursue computer science [15] and increases their Information Technology fluency, which includes their contemporary skills, fundamental grasp of concepts, and intellectual capabilities [10], [14]–[17].

However, researchers have also reported limitations with pair programming, such as scheduling difficulties and collocating pairs [18], student resistance to pairing [19], [20], and dependencies on peers' programming abilities [21]. Further, establishing a synergistic collaboration between pairs can be challenging as they might be affected by their relational interactions [22], [23] or may be predisposed to negative stereotypes [24].

One way to address these problems is by replacing one human member of the pair with a conversational agent to promote the human's coding practices and avoid stereotyped views. Further, current technologies are enabling more natural forms of human-machine interaction and potentially even dissolving the boundaries between human-human and human-machine interactions through the use of Conversational Agents (CAs) and Artificial Intelligence (AI) [25], [26]. We wanted to investigate how we can further dissolve these boundaries, especially in a programming context.

To investigate the design space of a pair programming CA, we conducted a remote pair programming study with 18 participants. Our goal with this study was to understand how humans behave in a programming context to reveal possible design guidelines for an agent. Based on research from Wagner [27], Zhao [28], and Edelman [29], teaching students to collaborate, communicate, and think critically and creatively is central for student success and aligns with their innate desire.

- *RQ1: What creativity strategies are used by pair programmers?*
- *RQ2: What communication styles are used during pair programming?*

Identifying which strategies and styles, if any, transfer matters because research suggests that human-agent and

978-1-7281-6901-9/20/$31.00 ©2020 IEEE

TABLE I
THE FOUR PHASES OF CPS AND THEIR DEFINITIONS, ALONG WITH EXAMPLES OF PARTICIPANTS IMPLEMENTING TIC-TAC-TOE.

Phases	**Definitions**	**Examples (from our study)**
Clarify: Explore the Vision	Identify the goal, wish, or challenge.	*"You need to implement the game features, such as checking if we have a winner"*
Clarify: Gather Data	Describe and generate data to enable a clear understanding of the challenge.	*"So I guess that's testing if someone put down a mark on the tile [referring to a test case in the code]."*
Clarify: Formulate Challenges	Sharpen awareness of the challenge and create questions that invite solutions.	*"We want to check if there are three vertical, horizontal, diagonal for the same mark."*
Idea	Generate ideas on how to solve the challenge	*"It's a three-by-three, so we could go from a loop for rows and then a loop for columns and then a loop that's testing diagonal."*
Develop	To move from ideas to solutions. Evaluate, strengthen, and select solutions for best "fit."	*"We would want to have a section of code that qualifies for checking the rows, and another for checking the columns. Just to structure the code and organize it."*
Implement	Explore acceptance and identify resources/actions that support implementation of the selected solution(s).	*"Change that, the board to x again. And then if we test that, vertical and horizontal come back as true. [implementing code that gives the desired result]"*

human-human interactions differ in many aspects and that mechanically applying existing theories of human-human interaction to the design of human-agent collaboration is ill-advised [30]–[32]. Thus, to investigate which creativity strategies and communication styles transfer to CA design, we conducted a follow-up Wizard of Oz study. By examining the set of transferable strategies and the set of non-transferable ones, we aim to identify general mechanisms that mediate how people sense, comprehend, and interact with artificial agents.

- *RQ3: Which creativity strategies and communication styles transfer from human-human interactions to human-agent interactions?*

II. BACKGROUND

A. Osborn-Parnes Creative Problem Solving

We leveraged the Osborn-Parnes Creative Problem Solving (CPS) Process [33]–[36] to identify different creativity strategies that pairs and individuals engage in. Table I shows the four phases of the CPS process. CPS is flexible, and its use depends on the situation. For example, if one already has a clearly defined problem, the process would begin at the idea phase. CPS has been used in computer science education to teach programming [37], [38]. Hence, CPS process is ideal to support a structured problem solving by a CA.

1) Pair-Programming Studies and Creativity: A pair's creativity increases with pair programming as Tanja et al. suggested that the solutions of the pairs should be better than those of individuals because the combined creativity and experience of both are greater [39]. Seo et al. found that software education using pair programming will be more effective on the development of elementary school students' creativity [40]. Begel and Nagappan noted creativity to be one of the top ten benefits of pair programming as reported by engineers from the industry [41]. Howard et al. found that students enjoyed programming and experienced more creativity in pairs than working solo [42]. None of these studies have done qualitative investigations of the creative problem solving process of programming pairs.

B. Communication - Dialogue Acts

In order to understand the pair of programmers' communication we used Dialogue Acts (DAs) which Andreas et al. defined DAs as "classified utterances according to a combination of pragmatic, semantic, and syntactic criteria" [43] (Table II). DAs are useful not only because they enable understanding of conversational speech and collaborative problem solving [17], [43], [44] but also enable understanding of conversations via a hidden Markov model (an AI algorithm) [43], a necessity for a CA.

1) Pair-Programming Studies and Communication: Da Silva et al.'s literature review on pair programming raised concerns about the inadequate number of studies related to remote pair programming and advocated for more studies on communication [45]. Thus we leveraged DAs to understand the verbal communication of pair programmers. The only study that investigated DAs in the context of pair programming is by Rodríguez et al. [17] which found that collaboration among undergraduate CS students is more effective when both partners make substantive dialogue contributions, express uncertainty, and resolve it. Our study differs as we investigate more DAs, use a remote pair programming setting, and focus on creating a CA.

C. Conversational Agents

Conversational Agents (CAs) are computer systems that are designed to interact with a human user through natural forms of conversations across a wide variety of domains. In education, Sklar and Richards [46] have categorized three agents to assist human learners: pedagogical agents, peer-learning agents, and demonstrative agents [47]–[52]. Educational conversational agents can teach or be taught by students, and serve as a learner companion to avoid the "isolation problem" of computer education. Here, we investigate the design space for CAs that can act as a peer instead of tutor or tutee.

In the software engineering domain, Wood et al. conducted a Wizard of Oz study and explored AI techniques to detect speech types in programmers questions/answers while repairing bugs [53]. We differ as we investigate pair programming, creativity strategies and communication styles,

TABLE II
THE DIALOGUE ACTS AND THEIR DEFINITIONS, ALONG WITH EXAMPLES OF PARTICIPANTS IMPLEMENTING A TIC-TAC-TOE GAME.

Dialogue Acts	**Definitions**	**Examples (from our study)**
Statement	A single declaration or remark.	*"I see there's already an x player."*
Acknowledgement	Acceptance of the existence of something	*"Okay."*
Agreement	Direct accordance in opinion	*"Yeah, sure."*
Indirect Instruction	Implicit or polite instruction	*"We just need to make a diagonal."*
YN Questions (QYN)	Questions asking for a yes/no answer	*"Did I close something?"*
YN Answers (AYN)	Yes/No answers	*"Yes."*
WH Question (QWH)	Question asking for who/what/when/where/why/how	*"What do I call this?"*
WH Answers (AWH)	Answering WH questions	*"That should be at the bottom."*
Direct Instruction	An Explicit instruction	*"Okay, and then go up to the top, and change that, the board to x again."*
Uncertainty	Dialogue that indicates uncertainty	*"That says if. Umm lets see. What is it called."*
Feedback	A response or comment expressing opinion/feeling	*"That's probably gonna be a better idea.", "That's too complicated."*
Apology	A regretful acknowledgment of failure	*"Oh sorry. Oh my gosh. I'm so sorry."*

and explore the transferable human-human aspects to a human-agent pair.

III. METHODOLOGY

To understand the human-human interactions in a pair programming context, we conducted a controlled lab study.

A. Participants

We selected 18 university students (9 men and 9 women) on a first-come, first-served basis while ensuring an equal gender[1] distribution. This was to avoid gender biases not only in the observational study, but also in our recommendations for CAs. Given recent findings over bias in AI tools, it is imperative to use gender-balanced data in machine learning to prevent the continuation of ideologies that disadvantage women [56]. All participants had basic object-oriented programming experience. The education level distribution of participants was 3 freshmen, 1 sophomore, 5 juniors, 7 seniors, and 2 masters students. We had 17 participants in the age range of 19-23 years and one in 30-40 years. Nine participants had two years of programming experience, 5 had three years experience, 3 had four years experience, and 1 greater than four years. Only six participants reported prior experience in pair programming. All participants were compensated with a $20 gift card for participating. Throughout the paper we refer to pairs with label PS# for same-gender and PX# for mixed-gender (i.e. PS4 denotes Pair 4, a same-gender pair). Participants are referred to with the following labels: PS2-M1 denotes M for man, 1 for 1st partner, in Pair 2. Likewise, PX3-W2 denotes in Mixed-gender (X) Pair 3, W for woman, 2 for 2nd partner in the pair.

B. Study Design

Since prior research has shown that there are differences between how same-gender and mixed-gender pairs collaborate, communicate and coordinate [57], 3 man-man, 3 woman-woman, and 3 man-woman pairs were formulated. To emulate a remote pair programming environment, participants were placed in separate rooms during the study. Compared to co-located pair programming, remote pair programming is known to be comparable for student performance, effective code quality, and productivity [58]–[60]. Furthermore, since we were interested in designing a CA, we decided the remote programming setup of sharing a computer screen and remotely communicating through the computer more closely mimics interactions with a CA.

Participants used Eclipse IDE [61] for programming in Java and TeamViewer [62] for remote collaboration. The participants communicated using text, voice, and video chats provided by TeamViewer.

At the start of the study, participants were asked to fill out a background questionnaire. They were also given a tutorial about pair programming and the think-aloud method [63]. All participants used the think-aloud method to vocalize their thoughts as they worked on their programming tasks. After being introduced to these concepts, participants were given a simple pair programming task to fix a validation test in a password checking implementation with the purpose of encouraging pair jelling. Pair jelling is a period of time when pairs first work together and become acquainted with the pairing, after which they perform considerably more efficiently [64], [65]. During the pair programming tasks, participants freely switched roles as necessary.

After the simple task, participants were given the main task which was to implement a Tic-Tac-Toe game within a 40 minute time period. The time limit served to ensure the entire session was below 90 minutes as well as to prevent participants from fatiguing. Facial reactions, audio, and screen interactions of participants were recorded through Morae [66], a screen capture tool. In the Tic-Tac-Toe game, the players take turns marking spaces in a 3x3 board. Participants had to add code and test cases to complete the task. Participants were provided with user stories and scenarios of the game, along with the implementation of the board and three related test cases. Tic-Tac-Toe was selected for its simplicity, as anyone with basic Java experience could understand and implement solutions for the requirements, without prior knowledge of the domain.

[1]We will only focus on the two most common genders: "women" and "men." As with [54], [55], gender is a person's gender identification. We use the term "men" as a shorthand for "people who identify as men" and "women" to denote "people who identify as women."

C. Analysis of Data

Video and audio of the participants were transcribed and qualitatively coded according to the stages of creativity (refer Table I) and communication Dialogue Acts (refer Table II) code sets. Codes were assigned to the 30-seconds segment split of the task, allowing multiple codes per segment. 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 [67]. Then the researchers split the remaining transcript data and coded it independently. The researchers used thematic analysis [68] to organize qualitative data into themes which relate back to the research questions.

IV. Results

Here, we answer our RQs and propose Design Guidelines (referred to as G#) for future CAs to serve as pair programming partners (refer Table III). The design guidelines followed a consistent format as recommended by Amershi et al. [69]. We used deductive and inductive reasoning to develop guidelines. Example: For G1, based on our results we deducted that participants verified after every stage of creativity and then using inductive reasoning formed G1.

A. RQ1: Pair Programming Creative Strategies

In order to answer RQ1, we leveraged the Osborn-Parnes Creative Problem Solving (CPS) to understand our participants (as discussed in Table I). Note, that our participants were not explicitly asked to follow the CPS process. Participants evenly fell across the four CPS phases (Clarify 25.6%; Idea 24.0%; Develop 22.7%; Implement 27.7%). We also observed that participants did not stay in one phase for extended periods of time as they completed tasks. Thus, no phase took significantly more time than another.

Overall Strategies as a Pair

Verifying after each creativity phase → G1 Pairs followed the CPS process with additional verification at the end of each phase. The modified model is shown in Figure 1. The pairs moved to the next phase of the creative process if they got positive responses from their partner, artifacts, or environment and in the case of a negative response, they backtracked to previous phases. For example, during the implement phase, some pairs would backtrack to the develop phase in order to rethink their logic and form a more effective solution. In PS2, PS2-M1 verified with his partner PS2-M2, *"our test passes on that, correct?"* which prompted PS2-M2 to run the test and realize *"the false one is not right"*. The pair then backtracked in their solution to fix the errors they discovered. We found that 5 pairs backtracked after verification.

Understand and elaborate ideas → G2 Eight out of our 18 participants employed visualization techniques to organize their thinking as well as to communicate thoughts and ideas about the program with each other. For instance, participant PS1-M2 commented, *"I'm going to write down an array on my sheet of paper... So we got something like this [shows the paper with drawing through camera]"* and used his drawing to explain his thoughts to his partner.

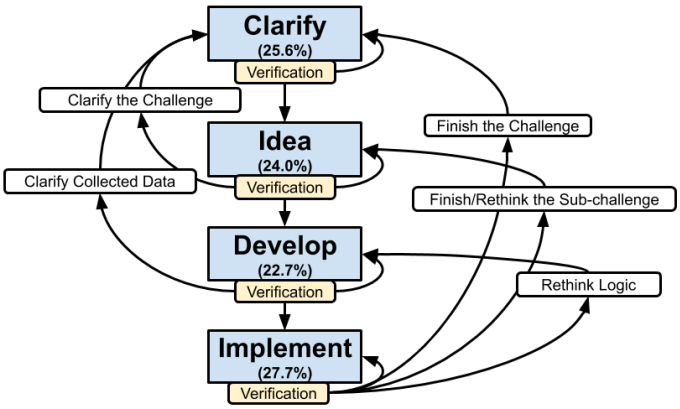

Fig. 1. Modified Creative Problem Solving process followed by pair programmers in our study. After each phase, participants verified with their partner, artifacts, or environment to move forward or backtrack within the process. The percentage of time spent in each phase by participants is shown with each phase label.

In addition to visualization, participants used a variety of strategies to organize and convey ideas. These included questions and clarifications (2), body language (2), and writing abstract code (1). For example, PS4-W1 asked, *"Should we do three functions for winner?"* By doing this, the participant was able to share and improve the idea based on her partner's response. Another participant, PS5-W1, employed body language to help show her ideas, stating *"they are on horizontal or one vertical or one diagonal"* while using hand gestures to convey her thoughts.

Participants used both breadth and depth-first approaches → G3 In breadth-first approach the participants formulated many ideas, and then discarded unnecessary ones. Seven out of 18 participants in our study used a breadth-first approach. For example, while considering what test cases were necessary, PX6-W2 stated, *"Let's go ahead and get rid of placeAnOMarkTest (a test case), we can always re-write it if we need to."* Shortly thereafter, the participant decided this case was not needed and deleted it.

In contrast, in depth-first approaches, participants formulated an idea and looked more deeply into it instead of considering alternatives. Seven out of 18 participants used a depth-first approach. PS9 exhibited a depth-first approach as the pair did not consider a different solution until one was fully fleshed out and implemented, as PS9-M1 commented, *"the other way to do it is..."* as he began to present an alternate solution, but eventually decided *"that's too complex"* and kept the already-made solution.

Thinking generally before tackling specifics to allow for organized problem-solving → G4 Five out of nine pairs utilized this strategy. PS2-M2 commented, *"We'll have to make two hypothetical games, or maybe a lot... We'll make a board object, fill it to some degree, and then check if the game is over."* He left the board state abstract, allowing the pair to plan what they needed to do when checking all

TABLE III
PROPOSED DESIGN GUIDELINES. GUIDELINES G1-G4 ARE FOR CREATIVITY AND G5-G12 FOR COMMUNICATION.

G#	Proposition
G1	**Support verification after every phase of creativity.** Ask for verification as a driver after completing the phase, and as a navigator, verify a human partner's verification requests.
G2	**Support ideas visually and verbally.** As a driver, the agent should be able to understand a human's ideas and as a navigator communicate the ideas.
G3	**Support breadth-first and depth-first idea generation.** The agent should be able to implement both idea generation approaches as a driver and understand them as a navigator.
G4	**Support problem solving via abstract thinking and specific tasks.** As a driver, the agent should be able to use both techniques and as a navigator should be able to recognize when the human uses them.
G5	**Clearly acknowledge suggestions.** The agent should acknowledge a human's suggestions as a driver and identify tasks or methods as a navigator.
G6	**Show clear agreement when a human partner is correct.** As a driver, the agent should positively acknowledge directions given by the human. As a navigator, if the human's ideas or implementations are correct, the agent should acknowledge them positively.
G7	**Give indirect instructions politely. Give direct instruction as a backseat driver.** The agent in driver/navigator role should provide polite indirect instructions. The agent should act as a backseat driver (on human's request) and give direct instructions.
G8	**The WH question's answers should be accompanied by directions and suggestions.** If the agent can not verify human's decisions it should politely ask WH questions to try and understand.
G9	**Clearly explain the reason for yes/no decisions.** The agent should answer in yes/no format and give an accompanying explanation for the reason why the answer is yes/no.
G10	**Give feedback related to the correctness and the completion of code with a positive tone.**
G11	**Express uncertainty with verbal and non-verbal cues.** As a driver, the agent can use filler words and embodiment or avatars to express uncertainty, and in the navigator role, the agent should be able to detect uncertainty, reassure and redirect humans to a correct solution.
G12	**Show apology when the answer is unknown or a mistake is made.** As a driver, the agent should apologize if it does not know the answer or made a mistake, while as a navigator, it should use empathizing dialogue when the human apologizes.

the ways to win across multiple cases. PS4-W2 commented, *"How about if we test true and false? ... so we are checking if there are three marks,"* when considering the general way to check for a win. By considering the general case instead of a specific instance, the participant created a plan to tackle the task.

In contrast to thinking generally, three pairs were observed to consider specific cases as a starting point. Focusing on a specific case allowed a pair to understand it well and then extend its ideas to other cases. An example is in PX6; the pair tried to implement a test that checks if the Tic-Tac-Toe game has a vertical win, as participant PX6-M1 commented, *"Let's say it's in the top left corner, and then the left most column is the winning one, let's say that's the case."* The participant picked a specific game situation that resulted in a win, the left column having three in a row, and used it to understand how the vertical win method should work. Focusing in on this specific case made the problem less abstract, potentially making it easier to see a solution.

B. RQ2: Pair Programming Communication Styles

In order to answer RQ2, we used Dialogue Acts (DAs) to understand the verbal communication between the pair of programmers (as discussed in Table II). Table IV shows the comprehensive list of DAs with their frequencies.

Conversational Style: Major Dialogue Acts used by Pairs

Statements were used most often, but they are less interesting, because they are the simplest DA and were used by participants for "descriptive, narrative, and personal" use, similar to findings of [43]. Other major DAs used were:

Acknowledgement → G5 Acknowledgement was used by all 18 participants as a short response to the identification of a new task or method of implementation typically via short responses such as *"yeah," "uh-huh," "alright,"* and *"okay"*.

Agreement → G6 All 18 participants used agreement as a verbal positive response to a question such as checking the validity of code, or a new idea such as how to implement a task. Agreements were elaborated upon more than acknowledgements when responding, such as how PX6-W2 said *"I think we should do that,"* when her partner proposed a new task.

TABLE IV
THE DIALOGUE ACT TAGS MOST PROMINENTLY FOUND IN OUR STUDY. DA FREQUENCIES ARE GIVEN AS USED BY MEN (# OF MEN PARTICIPANTS THAT USED THE DA), WOMEN(# OF WOMEN PARTICIPANTS THAT USED THE DA), THE TOTAL AMOUNT, AND THEIR PERCENTAGES OF THE TOTAL NUMBER OF UTTERANCES IN THE OVERALL CORPUS.

Dialogue Acts	M(#)	W(#)	Total	%age
Statement	766(9)	733(9)	1499	40.56%
Acknowledgement	223(9)	129(9)	352	9.52%
Agreement	119(9)	124(9)	243	6.57%
Indirect Instruction	118(9)	65(9)	183	4.95%
YN Question	84(9)	89(9)	173	4.68%
Feedback	88(9)	69(8)	157	4.25%
Direct Instruction	72(9)	69(9)	141	3.81%
Uncertainty	65(8)	71(9)	136	3.68%
YN Answer	49(9)	84(9)	133	3.60%
WH Question	45(9)	61(9)	106	2.87%
WH Answer	27(8)	21(8)	48	1.30%
Apology	8(6)	20(9)	28	0.76%

Indirect and direct instructions → G7 Indirect instruction was a polite or suggestive way of providing directions. All participants used them for taking and giving control, identifying tasks, running tests, suggestions, agreeing to tasks, reusing code, and giving an abstract view of the task or implementation. Directed instruction was used for similar reasons, but with directions given explicitly. For example, PX6-W2 told her partner *"Go up to the top, and change that— the board to x again."*

Asking and Answering WH questions → G8 All 18 participants used WH questions and only 16 responded with WH answers. WH questions were asked regarding topics such as implementation, deciding the next task, and deciding navigator/driver roles. We observed participants giving explanations or directions when asked WH questions.

Answering Yes/No questions → G9 All 18 participants asked and responded to yes/no questions. Yes/no questions were used for simpler topics such as the verification of ideas and code. Elaborations were also offered for example PS2-M1 asked *"So do we want the board to be full in that case?"* to which PS2-M2 answered *"It doesn't really matter. it should be a test for multiple case where there's no winner. So you can have a lot of different cases, you know."*

Feedback → G10 Seventeen participants used positive feedback for reasons such as correctness, affirmation, and completion. They also gave feedback when removing things such as *"I don't think we need it. Remove it."* as said by PS5-W2 in reference to a line of code.

Uncertainty → G11 Seventeen participants showed uncertainty when they made suggestions, responded to queries, decided navigator/driver roles, and determined the next task. When navigating, a participant's uncertainty was often verbalized with filler words like *"uhm"* and *"yeah"* or non-verbal ones such as laughter. For example PS2-M2 commented *"yeah, so. For the diagonal, we can just uhm."* When participants found themselves uncertain of what to do next, their partner would step in the driver role to assist. For example PS5-W1 commented, *"Uhm. I'm always confused with the for-loop. (PS5-W1 laughs)."* PS5-W2 then switched roles by saying *"Yes. Let me try something."* Hence, participants used non-verbal communication such as laughing and verbal communication with filler words to show uncertainty as known in communication theory [70].

Apology → G12 Fifteen participants, apologized when they were incorrect. For example PS5-W2 commented [referring to an array index], *"From three."* and was corrected by PS5-W1 who said, *"From two."*. PS5-W2 then apologized *"Yeah. Sorry."* Apology was sometimes accompanied by nonverbal communication such as hand gestures and laughter.

C. RQ3: Transferable Creativity Strategies and Communication Styles

To answer RQ3, we conducted a preliminary Wizard of Oz lab study to 1) investigate strategies transferable from human partnerships to agent partnerships and 2) identify effective strategies for agent design that deepen our understanding of the differences between human-human and human-agent interaction in a pair programming context.

The design of the study looked beyond current technology in which humans expect the agent to be of poor quality, and instead considered what such an interaction may require if the agent was as closely skilled to that of a human. This was the motivation for the wizard of oz design.

1) Preliminary Wizard of Oz study: A Wizard of Oz study is a rapid-prototyping method that examines interfaces that are either very technically demanding or are yet to be created [71]–[73]. Such studies help develop user-friendly interfaces that promote natural language dialogue and consider the unique qualities of man-machine interaction as distinct from general human discourse [74], such as studying user interactions with CAs [75]–[77].

Figure 2 is the snapshot of what our participants saw on their screen. The "agent" was implemented with a chat window using Saros plugin for Eclipse IDE [61], avatar using FACSvatar [78] and text-to-voice conversation using gTTS [79]. Participants had the option to change the arrangement of the programming and video chat windows, but none of the participants rearranged the windows. The wizard acting as an agent replaced text recognition and intent understanding, and used a protocol to implement the study. For example, when P1 said *"How to write code for win"* the intent for this was writing code, and the agent used the wizard protocol (detailed [80]) to respond *"I can make a recommendation from online, would you like me to do so?"* We used a constrained Wizard of Oz protocol [81] with a small set of available intents, so that the agent would exhibit a realistic level of intelligence to study programmer responses. If participants asked questions beyond the protocol, the wizard answered *"I'm afraid logic isn't my strong suit"* to simulate the behavior of a potentially automated system.

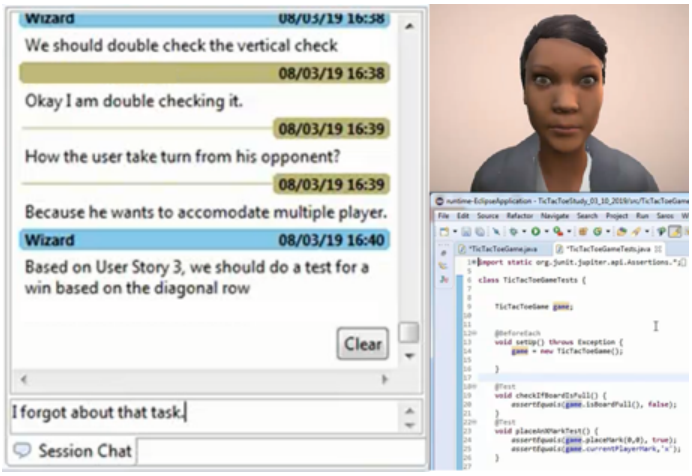

Fig. 2. Snapshot from Wizard of Oz study. The participants viewed facial expressions of agent (wizard) in the form of an avatar and interacted with it through a chat window and voice.

Participants: The human-agent study was conducted with seven university students (3 undergraduate, 3 graduate). We analyzed the data of 6 students (5 men and 1 woman) as one did not use the agent. Ages varied (3 were 19-23 years; 2 were 24-29 years; 1 was 30-40 years). Programming experience also varied (3 had less than a year of programming experience; 3 had more than four years of experience). None of the participants had prior pair programming experience. We compensated participants with a $20 gift card. Pairs are referred to as PB# (i.e. PB3 is pair 3) with human participants labeled as PB#-P1 and the agent as PB#-B1.

Study Design: The human-agent study was conducted with the same structure as the human-human study (tutorial and main task), with the exception of the pair-jelling simple task, as pairs had only one human participant. We also interviewed participants to get feedback about the agent design. Analysis used similar code sets as in the prior study.

Results: We investigated the transferability of design guidelines related to creativity strategies and communication

TABLE V
TRANSFERABILITY OF CREATIVITY (G1-G4) AND COMMUNICATION (G5-G12) GUIDELINES FROM HUMAN-HUMAN TO HUMAN-AGENT PAIR PROGRAMMING. THE PEACH COLORED ROWS SHOWS HARD TO TRANSFER GUIDELINES.

Design Guidelines	**Remarks**
G1 Verify	All 6 participants verified with the agent or another source. PB5- P1 took time to verify if code worked and commented, *"Okay, where is this going?"* The agent should detect each phase of creativity and use verification after each one, such as asking if the human partner is satisfied with the results e.g. ideas, code, tests.
G2 Visualization	Some participants preferred using visualization to organize thoughts. For example, PB2-P1 commented, *"what I need now is a pen and a paper. That is what usually I do."* The agent must be able to support this by using sketch-based whiteboard apps (such as [82]–[84]), though it will be difficult to have the agent generate ideas.
G3a Breadth-First	The participants in human-agent study did not use breadth-first approach when working with an agent. With current technology an agent won't be able to generate its own ideas but should be able to recognize it in a partner and can facilitate divergent thinking for partner. Graphics and interface design tools support divergent thinking by creating multiple options for users and supporting them to manipulate and compare those options [85]–[89].
G3b Depth-First	All participants opted to stick with their ideas and use a depth-first approach. The agent should be able to recognize when a human partner tries to use the depth-first approach. It should also be able to use the approach when ideas are being generated, though generating ideas itself may not be viable.
G4a Abstract Solutions	Participants did not opt to use abstract thinking with the agent, as help from it was more concrete. Designing the agent to think with abstraction will be difficult and unlikely to accomplish. However, recognizing when the human partner is trying to use abstraction could be viable.
G4b Specific Solutions	The agent should be able to consider specific cases when the human partner is attempting to use them to bolster understanding. For example, PB4-P1 commented, *"So now I'm going to be typing, um, so it would be three marks in a vertical column."*. Here agent can detect intent of the task.
G5 Acknowledgement	Our participants said in interviews that they like when the agent acknowledged and encouraged them. For example, PB5-P1 commented *"I liked the encouragement, that was cute."* Acknowledgement would be simple to implement when responding to ideas and suggestions, but testing the correctness of the ideas would be difficult.
G6 Agreement	CAs would have to be able to verify the correctness of any statement given by the human in order to agree, so unless the idea is closely related to auto-generated code or test cases it will be very hard to implement.
G7a Indirect Instr.	It is hard for a CA to auto-generate indirect instructions for a human conversation.
G7b Direct Instr.	When an agent can apply auto-generated code and test cases, it can act as a backseat driver. It will also be able to tell its human partner what should be done next. For example in PB7, PB7-B1 said *"we need to write a method to check for a horizontal win."*
G8 WH Question	Dialogue generation may be challenging for WH questions, especially if not related to auto-generated code or test cases. Ko et al. shows this is possible to an extent, as Whyline implements WH questions [90], [91].
G9 YN Question	For questions related to auto-generated code and test cases, it will be possible. For example PB1-B1 asked *"Is this code example from online useful?"* Questions regarding idea verification will be difficult however.
G10 Feedback	All 6 participants in our study appreciated the feedback given by the agent. For example, PB4-P1 commented, *"the partner feedback was pretty good."* An agent can give feedback for correct tests and code.
G11 Uncertainty	In presence of an agent participants had more uncertainty, for example, PB4-B1 commented *"we will need to place marks at zero two, one one, and two zero."* to which M1 replied *"ah, I didn't do, wait I didn't do all the unit tests. Ah, I feel like an idiot."* The agent should provide explanations to combat uncertainty.
G12 Apology	Participants in our study expected agent to apologize when it made mistakes or did not know the answer. For example, in an interview PB2-P1 commented *"if he apologized my trust would come back,"* after the agent gave flawed code. It should also be able to recognize apologies and respond reassuringly.

styles from a human-human to human-agent context.

Transferable Creativity Strategies. Table V contains design guidelines (G1-G4) discussed in RQ1 and how they can be transferred into a CA. Creativity strategies that require generating unique ideas, such as breadth-first idea generation and use of abstract thinking, are harder to transfer so cannot be fully realized by an agent. Instead, the agent will function as a synergistic partner supporting a human partner's ability to do these things.

Transferable Communication Styles. Table V shows the transferable design guidelines (G5 to G12) from human-human study to human-agent. Some of the designs are harder to transfer to a CA because humans are good at solutions, ideas, and designs relative to computers. It is possible to automatically generate source code and test cases as well as classify intent based on DAs to an extent. Hence, designs relying on these factors can be implemented by a CA in a driver/navigator role.

The dialogue templates for CAs should be designed to engage programmers by providing non-authoritative suggestions [92] and motivate the partner following Neilsen's Heuristic [93]. All 6 participants indicated in their interview that they liked and preferred the agent motivating them and verifying their implementations. For instance, PB5-P1 commented *"I like the response... when I was correct and where I was wrong"*.

Interruptions by partner. Interrupting the human partner only when necessary and relevant to the current topic can be a transferable aspect. In the human-human study, participants said they did not want to interrupt their partner as they considered it to be impolite, had low confidence, or did not want to be distracting. In the human-agent study, we found that humans interrupted the agent when they did not know what to do next, were stuck, were unsure about their problems, or wanted clarification. The agent would interrupt whenever it had a suggestion, occasionally frustrating the participant leading them to mute or ignore the suggestions. Similar to human-robot collaboration [94], programmers ignored an incorrect problem-solving strategy given by an agent. This may suggest an agent should not interrupt a human until there is a high error probability and that, for minor issues, it should wait until the human

has finished the current task.

V. Limitations

One limitation of this study is the small sample size of 9 pairs in the human-human study and 6 pairs in the human-agent study. Further, the human-agent study did not have a gender balance, although the transfer of strategies was independent of gender as the guidelines were gender-inclusive. Additionally, being a think-aloud study, DAs may not reflect real life communication scenarios. Consequently, Table IV may not accurately represent the true communication between the human participants, but it may indicate a trend for how dialogue acts are used. Our study also chose the intents for the CA trying to mimic what may be considered reasonable given the current state of research. However, AI is growing and it is difficult to guess the where technology will be when such an agent is created. Similarly, the level of integration between the agent and the code determines what functionality an agent can provide as a partner and likewise had to be estimated for RQ3. An actual agent may have more or less features available depending on its integration with the codebase.

VI. Discussion

To create a CA for pair programming, we need to parse the intent of utterances automatically for creativity phases or DAs. Such research has many challenges since the dialogue between programmers differs greatly from everyday conversations. Further, most research in natural language processing is focused on the physical world where topics are more discrete and universally established, while software artifacts are not and vary by their use.

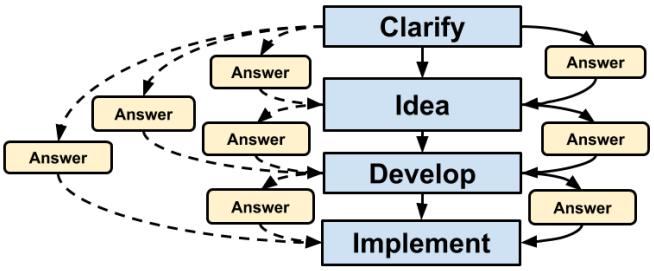

Fig. 3. Shows the human-human and human-agent interactions. (Left dotted black arrows) shows how humans skipped idea and develop phase when provided with answers (code/test cases) from an agent. (Right black arrows) shows humans pairs creatively programming.

In our human-agent study, we discovered potential issues with our primitive design for a CA and how it affected the CPS process. In the human-human study, participants thoroughly followed the creativity process, but in the presence of an agent, human-agent participants skipped over idea generation and jumped straight to implementing code (refer to Figure 3) as they got code and test case templates from the agent. This was evident as the code frequencies in the idea phase dropped from 24% to 14.2%; additionally, the implement phase code frequencies increased by 7.5%. Our results suggest that working with the agent caused participants to spend less time in idea generation and thus their creative output was lower. We recommend designers and practitioners be cautious when designing CAs to support programming, especially while handling the creative process, as to not hinder idea generation.

A. Implications for Pair Programming CA

In addition to the design guidelines discussed in the results section, and for Human-AI interactions [69], we recommend the following for a pair programming CA:

1) Including the Social Intelligence: CA for pair programming needs to exhibit social intelligence through effective floor management (turn-taking) by supporting features such as actively listening and sensing to determine who is talking and when to interject, detecting user confusion, sensing disengagement, etc. Further, by continuously "being there" and making rapport with the programmer, an agent may create "social influence" which has positive effects on persuasion and trust. In our human-agent study, four participants mentioned that they liked the social aspect "being there" of the agent.

2) Including Embodiments: CAs may include the optional feature of face and voice. Embodiments are known to improve trustworthiness, interaction engagement, user perception, and task performance [75], [95]–[97], especially for tasks that require continuous engagement like pair programming. An embodiment may offer unique benefits, such as during an interview, when PB4-P1, who had a hearing disability, expressed appreciation for the avatar and mentioned as he could supplement his hearing with lip reading. Yet, empirical evidence is mixed about the necessity of the embodiment [98]–[103], thus we propose offering users an option to disable it.

3) Including Trustable Traits: Irrespective of gender, trust is built between human partners over time with an open-minded, polite, confident and knowledgeable partner [57]. Based on existing literature, humans trust agents easily [104]. Further, in our human-agent study, participants trusted the agent without question. In one case, PB7-P1 responded, *"I'm going to blindly believe you."* Thus, an agent should be designed to be confident, transparent in its decisions, and to show its vulnerabilities (limitations as a machine) to increase the trust with a human partner over time.

VII. Conclusion

This is the first research to consider the design space for a pair programming conversational agent. Our in-depth qualitative analysis revealed human-human pair programmer's creativity strategies and communication styles resulting in 12 design guidelines for designing a conversational agent. Further, our human-agent wizard of oz study identified which of these are transferable to an agent. Such differences are not only important from a design perspective but also are critical for programmers who develop conversational agent technologies. With this foundational work, we take the first steps in making a conversational agent pair programmer one step closer to reality.

References

[1] A. Woolley, I. Aggarwal, T. Malone, Collective intelligence and group performance, Current Directions in Psychological Science 24 (2015) 420–424. doi:10.1177/0963721415599543.

[2] D. W. Palmieri, Knowledge management through pair programming, 2002.

[3] 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, 2003. ISESE 2003. Proceedings., 2003, pp. 143–152. doi:10.1109/ISESE.2003.1237973.

[4] L. Williams, R. Kessler, Pair Programming Illuminated, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[5] C. Barra, B. Crawford, Fostering creativity thinking in agile software development, Vol. 4799, 2007, pp. 415–426. doi:10.1007/978-3-540-76805-0_37.

[6] A. Belshee, Promiscuous pairing and beginner's mind: Embrace inexperience, 2005, pp. 125 – 131. doi:10.1109/ADC.2005.37.

[7] A. Cockburn, L. Williams, Extreme programming examined, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001, Ch. The Costs and Benefits of Pair Programming, pp. 223–243.
URL http://dl.acm.org/citation.cfm?id=377517.377531

[8] T. DeMarco, T. Lister, Peopleware: Productive Projects and Teams, Dorset House Publishing Co., Inc., New York, NY, USA, 1987.

[9] 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. doi:10.1145/2652524.2652529.
URL http://doi.acm.org/10.1145/2652524.2652529

[10] 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, SIGCSE '02, ACM, New York, NY, USA, 2002, pp. 38–42. doi:10.1145/563340.563353.
URL http://doi.acm.org/10.1145/563340.563353

[11] 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) (2004) 7–11. doi:10.1145/1028174.971307.
URL http://doi.acm.org/10.1145/1028174.971307

[12] 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, 2003, pp. 602–607.
URL http://dl.acm.org/citation.cfm?id=776816.776899

[13] L. A. Williams, E. N. Wiebe, K. Yang, M. Ferzli, C. Miller, In support of pair programming in the introductory computer science course, Computer Science Education 12 (2002) 197–212.

[14] 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 '16, ACM, New York, NY, USA, 2016, pp. 90–95. doi:10.1145/2839509.2844558.
URL http://doi.acm.org/10.1145/2839509.2844558

[15] L. L. Werner, B. Hanks, C. McDowell, Pair-programming helps female computer science students, J. Educ. Resour. Comput. 4 (1) (Mar. 2004). doi:10.1145/1060071.1060075.
URL http://doi.acm.org/10.1145/1060071.1060075

[16] 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, SIGCSE '18, ACM, New York, NY, USA, 2018, pp. 771–776. doi:10.1145/3159450.3159516.
URL http://doi.acm.org/10.1145/3159450.3159516

[17] 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, SIGCSE '17, ACM, New York, NY, USA, 2017, pp. 507–512. doi:10.1145/3017680.3017748.
URL http://doi.acm.org/10.1145/3017680.3017748

[18] B. F. Hanks, Distributed pair programming: An empirical study, in: C. Zannier, H. Erdogmus, L. Lindstrom (Eds.), Extreme Programming and Agile Methods - XP/Agile Universe 2004, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 81–91.

[19] A. Oram, G. Wilson, Making Software: What Really Works, and Why We Believe It, 1st Edition, O'Reilly Media, Inc., 2010.

[20] L. Williams, B. Kessler, 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, Washington, DC, USA, 2000, pp. 59–.
URL http://dl.acm.org/citation.cfm?id=794188.794326

[21] H. Gallis, E. Arisholm, A transition from partner programming to pair programming-an industrial case study, 2002.

[22] C. Berger, R. Calabrese, "some exploration in initial interaction and beyond: Toward a developmental theory of interpersonal communication.", Human Communication Research 1 (2006) 99 – 112. doi:10.1111/j.1468-2958.1975.tb00258.x.

[23] E. P. Torrance, Influence of dyadic interaction on creative functioning., Psychological reports 26 2 (1970) 391–4.

[24] H. J. Spiers, B. C. Love, M. E. Le Pelley, C. E. Gibb, R. A. Murphy, Anterior temporal lobe tracks the formation of prejudice, J. Cognitive Neuroscience 29 (3) (2017) 530–544. doi:10.1162/jocn_a_01056.
URL https://doi.org/10.1162/jocn_a_01056

[25] D. Perez-Marin, I. Pascual-Nieto, Conversational Agents and Natural Language Interaction: Techniques and Effective Practices, Information Science Reference - Imprint of: IGI Publishing, Hershey, PA, 2011.

[26] S. Diederich, A. Brendel, L. Kolbe, On conversational agents in information systems research: Analyzing the past to guide future work, 2019.

[27] T. Wagner, R. A. Compton, Creating innovators: The making of young people who will change the world, Simon and Schuster, 2012.

[28] Y. Zhao, World class learners: Educating creative and entrepreneurial students, Corwin Press, 2012.

[29] B. Edelman, Inc, Creativity and education: Why it matters (2010).
URL http://www.adobe.com/aboutadobe/pressroom/pdfs/Adobe_Creativity_and_Education_Why_It_Matters_study.pdf

[30] M. F. Jung, J. J. Lee, N. DePalma, S. O. Adalgeirsson, P. J. Hinds, C. Breazeal, Engaging robots: Easing complex human-robot teamwork using backchanneling, in: Proceedings of the 2013 Conference on Computer Supported Cooperative Work, CSCW '13, ACM, New York, NY, USA, 2013, pp. 1555–1566. doi:10.1145/2441776.2441954.
URL http://doi.acm.org/10.1145/2441776.2441954

[31] B. Reeves, C. Nass, The media equation: How people treat computers, television, and new media like real people and places, Bibliovault OAI Repository, the University of Chicago Press (01 1996).

[32] L. Swartz, B. A. Nardi, Why people hate the paperclip: Labels, appearance, behavior, and social responses to user interface agents, Tech. rep. (2003).

[33] S. Isaksen, D. Treffinger, Celebrating 50 years of reflective practice: Versions of creative problem solving, The Journal of Creative Behavior 38 (06 2004). doi:10.1002/j.2162-6057.2004.tb01234.x.

[34] A. F. Osborn, Applied imagination : principles and procedures of creative thinking / by Alex F. Osborn, C. Scribner New York, 1953.

[35] M. Page-Jones, The Practical Guide to Structured Systems Design: 2Nd Edition, Yourdon Press, Upper Saddle River, NJ, USA, 1988.

[36] S. Parnes, Creative behavior workbook, Scribner, New York, 1967.

[37] R. E. Mayer, The psychology of how novices learn computer programming, ACM Comput. Surv. 13 (1) (1981) 121–141. doi:10.1145/356835.356841.
URL http://doi.acm.org/10.1145/356835.356841

[38] D. N. Perkins, F. Martin, Fragile knowledge and neglected strategies in novice programmers, in: Papers Presented at the First Workshop on Empirical Studies of Programmers on Empirical Studies of Programmers, Ablex Publishing Corp., Norwood, NJ, USA, 1986, pp. 213–229.
URL http://dl.acm.org/citation.cfm?id=21842.28896

[39] T. Bipp, A. Lepper, D. Schmedding, Pair programming in software development teams - an empirical study of its benefits, Inf. Softw. Technol. 50 (3) (2008) 231–240. doi:10.1016/j.infsof.2007.05.006.
URL http://dx.doi.org/10.1016/j.infsof.2007.05.006

[40] Y.-H. Seo, J.-H. Kim, Analyzing the effects of coding education through pair programming for the computational thinking and creativity of

elementary school students, Indian Journal of Science and Technology 9 (12 2016). doi:10.17485/ijst/2016/v9i46/107837.

[41] A. Begel, N. Nagappan, Pair programming: what's in it for me?, in: ESEM '08: Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, esem '08: proceedings of the second acm-ieee international symposium on empirical software engineering and measurement Edition, ACM, 2008, pp. 120–128.
URL https://www.microsoft.com/en-us/research/publication/pair-programming-whats-in-it-for-me/

[42] E. Howard, D. Evans, J. Courte, C. Bishop-Clark, A qualitative look at alice and pair-programming, Number 7 (08 2009).

[43] A. Stolcke, N. Coccaro, R. Bates, P. Taylor, C. Van Ess-Dykema, K. Ries, E. Shriberg, D. Jurafsky, R. Martin, M. Meteer, Dialogue act modeling for automatic tagging and recognition of conversational speech, Comput. Linguist. 26 (3) (2000) 339–373. doi:10.1162/089120100561737.
URL https://doi.org/10.1162/089120100561737

[44] A. Vail, K. Boyer, Identifying effective moves in tutoring: On the refinement of dialogue act annotation schemes, 2014, pp. 199–209. doi:10.1007/978-3-319-07221-0_24.

[45] J. Chong, T. Hurlbutt, The social dynamics of pair programming, in: Proceedings of the 29th International Conference on Software Engineering, ICSE '07, IEEE Computer Society, Washington, DC, USA, 2007, pp. 354–363. doi:10.1109/ICSE.2007.87.
URL https://doi.org/10.1109/ICSE.2007.87

[46] E. Sklar, D. Richards, The use of agents in human learning systems, in: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS '06, ACM, New York, NY, USA, 2006, pp. 767–774. doi:10.1145/1160633.1160768.
URL http://doi.acm.org/10.1145/1160633.1160768

[47] J. A. R. Uresti, Should i teach my computer peer? some issues in teaching a learning companion, in: G. Gauthier, C. Frasson, K. VanLehn (Eds.), Intelligent Tutoring Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2000, pp. 103–112.

[48] J. Ramirez Uresti, B. Du Boulay, Expertise, motivation and teaching in learning companion systems., International Journal of Artificial Intelligence in Education 14 (2004) 193–231.

[49] T.-W. Chan, Learning companion systems, social learning systems, and the global social learning club, Journal of Artificial Intelligence in Education 7 (2) (1996) 125.
URL https://www.learntechlib.org/p/82394

[50] L. Sheremetov, A. G. Arenas, Eva: an interactive web-based collaborative learning environment, Computers & Education 39 (2) (2002) 161 – 182. doi:https://doi.org/10.1016/S0360-1315(02)00030-1.
URL http://www.sciencedirect.com/science/article/pii/S0360131502000301

[51] Woolf, Copyright page, in: Building Intelligent Interactive Tutors, Morgan Kaufmann, San Francisco, 2009, p. iii. doi:https://doi.org/10.1016/B978-0-12-373594-2.00016-2.
URL http://www.sciencedirect.com/science/article/pii/B9780123735942000162

[52] K.-W. Han, E. Lee, Y. Lee, The impact of a peer-learning agent based on pair programming in a programming course, Education, IEEE Transactions on 53 (2010) 318 – 327. doi:10.1109/TE.2009.2019121.

[53] A. Wood, P. Rodeghero, A. Armaly, C. McMillan, Detecting speech act types in developer question/answer conversations during bug repair, in: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2018, ACM, New York, NY, USA, 2018, pp. 491–502. doi:10.1145/3236024.3236031.
URL http://doi.acm.org/10.1145/3236024.3236031

[54] C. WEST, D. H. ZIMMERMAN, Doing gender, Gender & Society 1 (2) (1987) 125–151. doi:10.1177/0891243287001002002.
URL https://doi.org/10.1177/0891243287001002002

[55] J. Butler, Revisiting bodies and pleasures, Theory, Culture & Society 16 (2) (1999) 11–20. doi:10.1177/02632769922050520.
URL https://doi.org/10.1177/02632769922050520

[56] S. Leavy, 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, GE '18, ACM, New York, NY, USA, 2018, pp. 14–16. doi:10.1145/3195570.3195580.
URL http://doi.acm.org/10.1145/3195570.3195580

[57] S. Kuttal, K. Gerstner, A. Bejarano, Remote pair-programming in online cs education: Investigating through a gender lens, IEEE Symposium on Visual Languages & Human-Centric Computing (2019).

[58] P. Baheti, D. Gehringer, P. Stotts, Exploring the efficacy of distributed pair programming, 2002. doi:10.1007/3-540-45672-4_20.

[59] R. Duque, C. Bravo, 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, 2008, pp. 270–276. doi:10.1109/ICSEA.2008.82.
URL https://doi.org/10.1109/ICSEA.2008.82

[60] B. Hanks, Empirical evaluation of distributed pair programming, International Journal of Human-Computer Studies 66 (2008) 530–544. doi:10.1016/j.ijhcs.2007.10.003.

[61] Eclipse IDE (2019).
URL https://www.eclipse.org/

[62] TeamViewer (2019).
URL https://www.teamviewer.com/en-us/

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

[64] D. Jones, S. Fleming, What use is a backseat driver? a qualitative investigation of pair programming, 2013, pp. 103–110. doi:10.1109/VLHCC.2013.6645252.

[65] L. Williams, R. Kessler, W. Cunningham, R. Jeffries, Strengthening the case for pair-programming, Software, IEEE 17 (2000) 19 – 25. doi:10.1109/52.854064.

[66] Morae (2019).
URL http://www.techsmith.com/morae.asp

[67] P. Jaccard, Etude de la distribution florale dans une portion des alpes et du jura, Bulletin de la Societe Vaudoise des Sciences Naturelles 37 (1901) 547–579. doi:10.5169/seals-266450.

[68] V. Braun, V. Clarke, Using thematic analysis in psychology, Qualitative research in psychology 3 (2006) 77–101. doi:10.1191/1478088706qp063oa.

[69] S. Amershi, D. Weld, M. Vorvoreanu, A. Fourney, B. Nushi, P. Collisson, J. Suh, S. Iqbal, P. N. Bennett, K. Inkpen, J. Teevan, R. Kikin-Gil, E. Horvitz, Guidelines for human-ai interaction, in: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, CHI '19, ACM, New York, NY, USA, 2019, pp. 3:1–3:13. doi:10.1145/3290605.3300233.
URL http://doi.acm.org/10.1145/3290605.3300233

[70] R. L. West, Introducing Communication Theory: Analysis and Application, McGraw-Hill, Inc., New York, NY, USA, 2007.

[71] L. Paul Green, THE WIZARD OF OZ: A TOOL FOR RAPID DEVELOPMENT OF USER INTERFACES.
URL https://books.google.com/books?id=XhWEpS1OooAC

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

[73] J. Wilson, D. Rosenberg, Rapid prototyping for user interface design, in: Handbook of human-computer interaction, Elsevier, 1988, pp. 859–875.

[74] N. Dahlbäck, A. Jönsson, L. Ahrenberg, Wizard of oz studies—why and how, Knowledge-based systems 6 (4) (1993) 258–266.

[75] T. Bickmore, J. Cassell, Relational agents: A model and implementation of building user trust, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '01, ACM, New York, NY, USA, 2001, pp. 396–403. doi:10.1145/365024.365304.
URL http://doi.acm.org/10.1145/365024.365304

[76] J. Bradley, D. Benyon, O. Mival, N. Webb, Wizard of oz experiments and companion dialogues, in: Proceedings of the 24th BCS Interaction Specialist Group Conference, British Computer Society, 2010, pp. 117–123.

[77] P. Wargnier, G. Carletti, Y. Laurent-Corniquet, S. Benveniste, P. Jouvelot, A.-S. Rigaud, 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, 2016, pp. 1–8.

[78] FACSvatar (2018).
URL https://github.com/NumesSanguis/FACSvatar

[79] Google text-to-speech python library (2019).
URL https://github.com/pndurette/gTTS

[80] Human-Agent study protocol is available at (2020).
URL https://docs.google.com/document/d/16ARnq3vX8bM4Flk7keKAy7sfeC57_jZTCGHuThTO6BU/edit?usp=sharing

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

[82] Explain Everything (2019).
URL https://explaineverything.com/

[83] LiveBoard (2019).
URL https://liveboard.online/

[84] RealTimeBoard (2019).
URL https://realtimeboard.com/

[85] M. Terry, E. D. Mynatt, Side views: Persistent, on-demand previews for open-ended tasks, in: Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology, UIST '02, ACM, New York, NY, USA, 2002, pp. 71–80. doi:10.1145/571985.571996.
URL http://doi.acm.org/10.1145/571985.571996

[86] M. Terry, E. D. Mynatt, K. Nakakoji, Y. Yamamoto, Variation in element and action: Supporting simultaneous development of alternative solutions, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '04, ACM, New York, NY, USA, 2004, pp. 711–718. doi:10.1145/985692.985782.
URL http://doi.acm.org/10.1145/985692.985782

[87] B. Hartmann, L. Yu, A. Allison, Y. Yang, S. Klemmer, Design as exploration: Creating interface alternatives through parallel authoring and runtime tuning, 2008, pp. 91–100. doi:10.1145/1449715.1449732.

[88] B. Hartmann, S. Follmer, A. Ricciardi, T. Cardenas, S. R. Klemmer, D.note: Revising user interfaces through change tracking, annotations, and alternatives, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '10, ACM, New York, NY, USA, 2010, pp. 493–502. doi:10.1145/1753326.1753400.
URL http://doi.acm.org/10.1145/1753326.1753400

[89] R. Kumar, J. O. Talton, S. Ahmad, S. R. Klemmer, Bricolage: Example-based retargeting for web design, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '11, ACM, New York, NY, USA, 2011, pp. 2197–2206. doi:10.1145/1978942.1979262.
URL http://doi.acm.org/10.1145/1978942.1979262

[90] A. J. Ko, B. A. Myers, Designing the whyline: A debugging interface for asking questions about program behavior, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '04, ACM, New York, NY, USA, 2004, pp. 151–158. doi:10.1145/985692.985712.
URL http://doi.acm.org/10.1145/985692.985712

[91] A. J. Ko, B. A. Myers, Finding causes of program output with the java whyline, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '09, ACM, New York, NY, USA, 2009, pp. 1569–1578. doi:10.1145/1518701.1518942.
URL http://doi.acm.org/10.1145/1518701.1518942

[92] D. Tannen, THE POWER of Talk: Who Gets Heard and why, HBR Onpoint [Harvard Business Review], Harvard Business Review, 2005.
URL https://books.google.com/books?id=RaPLswEACAAJ

[93] J. Nielsen, R. Molich, Heuristic evaluation of user interfaces, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '90, ACM, New York, NY, USA, 1990, pp. 249–256. doi:10.1145/97243.97281.
URL http://doi.acm.org/10.1145/97243.97281

[94] K. L. Koay, D. S. Syrdal, M. L. Walters, K. Dautenhahn, Five weeks in the robot house – exploratory human-robot interaction trials in a domestic setting, in: 2009 Second International Conferences on Advances in Computer-Human Interactions, 2009, pp. 219–226. doi:10.1109/ACHI.2009.62.

[95] A. Shamekhi, Q. V. Liao, D. Wang, R. K. E. Bellamy, T. Erickson, 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, CHI '18, ACM, New York, NY, USA, 2018, pp. 391:1–391:13. doi:10.1145/3173574.3173965.
URL http://doi.acm.org/10.1145/3173574.3173965

[96] J. Gratch, N. Wang, J. Gerten, E. Fast, R. Duffy, Creating rapport with virtual agents, in: C. Pelachaud, J.-C. Martin, E. André, G. Chollet, K. Karpouzis, D. Pelé (Eds.), Intelligent Virtual Agents, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 125–138.

[97] A. Takeuchi, T. Naito, Situated facial displays: Towards social interaction, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '95, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1995, pp. 450–455. doi:10.1145/223904.223965.
URL http://dx.doi.org/10.1145/223904.223965

[98] S. v. Mulken, E. André, J. Müller, 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, 1999, pp. 152–156.
URL http://dl.acm.org/citation.cfm?id=647944.741893

[99] S. van Mulken, E. André, J. Müller, The persona effect: How substantial is it?, in: H. Johnson, L. Nigay, C. Roast (Eds.), People and Computers XIII, Springer London, London, 1998, pp. 53–66.

[100] D. Hasegawa, J. Cassell, K. Araki, The role of embodiment and perspective in direction-giving systems (01 2010).

[101] R. Häuslschmid, M. von Bülow, B. Pfleging, A. Butz, Supportingtrust in autonomous driving, in: Proceedings of the 22Nd International Conference on Intelligent User Interfaces, IUI '17, ACM, New York, NY, USA, 2017, pp. 319–329. doi:10.1145/3025171.3025198.
URL http://doi.acm.org/10.1145/3025171.3025198

[102] D. M. Dehn, S. van Mulken, The impact of animated interface agents: A review of empirical research, Int. J. Hum.-Comput. Stud. 52 (1) (2000) 1–22. doi:10.1006/ijhc.1999.0325.
URL http://dx.doi.org/10.1006/ijhc.1999.0325

[103] N. Yee, J. N. Bailenson, K. Rickertsen, 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, CHI '07, ACM, New York, NY, USA, 2007, pp. 1–10. doi:10.1145/1240624.1240626.
URL http://doi.acm.org/10.1145/1240624.1240626

[104] T. Kessler, K. Stowers, J. Brill, P. Hancock, Comparisons of human-human trust with other forms of human-technology trust, Proceedings of the Human Factors and Ergonomics Society Annual Meeting 61 (1) (2017) 1303–1307. arXiv:https://doi.org/10.1177/1541931213601808, doi:10.1177/1541931213601808.
URL https://doi.org/10.1177/154193121360180