How Far Are We? The Triumphs and Trials of Generative AI in Learning Software Engineering

Rudrajit Choudhuri
Oregon State University
Corvallis, OR, USA
choudhru@oregonstate.edu

Dylan Liu
Oregon State University
Corvallis, OR, USA
liudy@oregonstate.edu

Igor Steinmacher
Northern Arizona University
Flagstaff, AZ, USA
igor.steinmacher@nau.edu

Marco Gerosa
Northern Arizona University
Flagstaff, AZ, USA
marco.gerosa@nau.edu

Anita Sarma
Oregon State University
Corvallis, OR, USA
anita.sarma@oregonstate.edu

ABSTRACT

Conversational Generative AI (convo-genAI) is revolutionizing Software Engineering (SE) as engineers and academics embrace this technology in their work. However, there is a gap in understanding the current potential and pitfalls of this technology, specifically in supporting students in SE tasks. In this work, we evaluate through a between-subjects study (N=22) the effectiveness of ChatGPT, a convo-genAI platform, in assisting students in SE tasks. Our study did not find statistical differences in participants' productivity or self-efficacy when using ChatGPT as compared to traditional resources, but we found significantly increased frustration levels. Our study also revealed 5 distinct faults arising from violations of Human-AI interaction guidelines, which led to 7 different (negative) consequences on participants.

CCS CONCEPTS

• **Human-centered computing → Empirical studies in HCI**.

KEYWORDS

Empirical Study, Software Engineering, Generative AI, ChatGPT

ACM Reference Format:
Rudrajit Choudhuri, Dylan Liu, Igor Steinmacher, Marco Gerosa, and Anita Sarma. 2024. How Far Are We? The Triumphs and Trials of Generative AI in Learning Software Engineering. In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE '24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3597503.3639201

1 INTRODUCTION

The advent of Conversational Generative AI (convo-genAI) is proving to be a "Gutenberg moment" across education and business, and software engineering is no exception. Convo-genAI systems (e.g., ChatGPT [66], Google Bard [42], Meta LLaMA [61]) generate novel content, be it a haiku or a relevant code snippet, informed by pre-existing data and minimal input. Additionally, these tools provide a conversational interface, enabling users to interact using natural language to generate outputs fine-tuned to their needs. Discussions on the use of generative AI (genAI) range from how it signals the "end of programming" [92, 98] to how it can transform software engineering for the better [29, 57].

Given the nascency of this innovation, it remains unclear how it can be leveraged in education, and uncertainty overshadows its potential benefits. The use of both conversational agents and genAI has been investigated in the context of Introductory Computer Science (CS). Prior work on conversational agents has demonstrated their effectiveness for non-specialists and learners, as they facilitate dialogue with an "expert" [40, 44, 86, 89]. For example, Loksa et al. [56] found that high-schoolers in an introductory programming summer camp benefited when they could seek help by articulating their problem-solving strategies and their current task state. Further, the new generation of students prefers talking with chatbots over talking to a person [2]. Indeed, there has been extensive work on conversational agents for education [65, 94] focusing on students' engagement [15, 39], self-directed learning [71], and tutoring [84]. GenAI systems have also been explored in this context [32, 59, 87]. But these studies have either been focused on using genAI to solve algorithmic problems [70, 83] (i.e., programming) or on improving genAI technology and output [52, 93] (i.e., better AI).

Currently, there is a gap in understanding how convo-genAI systems can be leveraged for learning advanced CS topics, such as Software Engineering (SE), where the learning objectives include unique complexities and contextual decision-making involving subjectivity and multiple trade-offs [72, 73]. In fact, SE education extends beyond the confines of classroom education, into the workplace, where software developers need to learn job-relevant topics, processes, and tools [13]. Learning in these situations requires contextualized assistance. Given that convo-genAI systems are capable of providing such assistance, coupled with natural language dialogue capabilities, they can be particularly well suited for SE topics [29], where traditional chatbots have fallen short so far [35]).

Thus, in this paper, we investigate: ***(RQ1): How effective is convo-genAI in helping students in software engineering tasks?*** We answered this research question through a between-subjects user study, where the Experimental group solely used ChatGPT, while participants in the Control group could use any resource other than genAI tools. We selected ChatGPT as a representative

ICSE '24, April 14–20, 2024, Lisbon, Portugal
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0217-4/24/04.
https://doi.org/10.1145/3597503.3639201

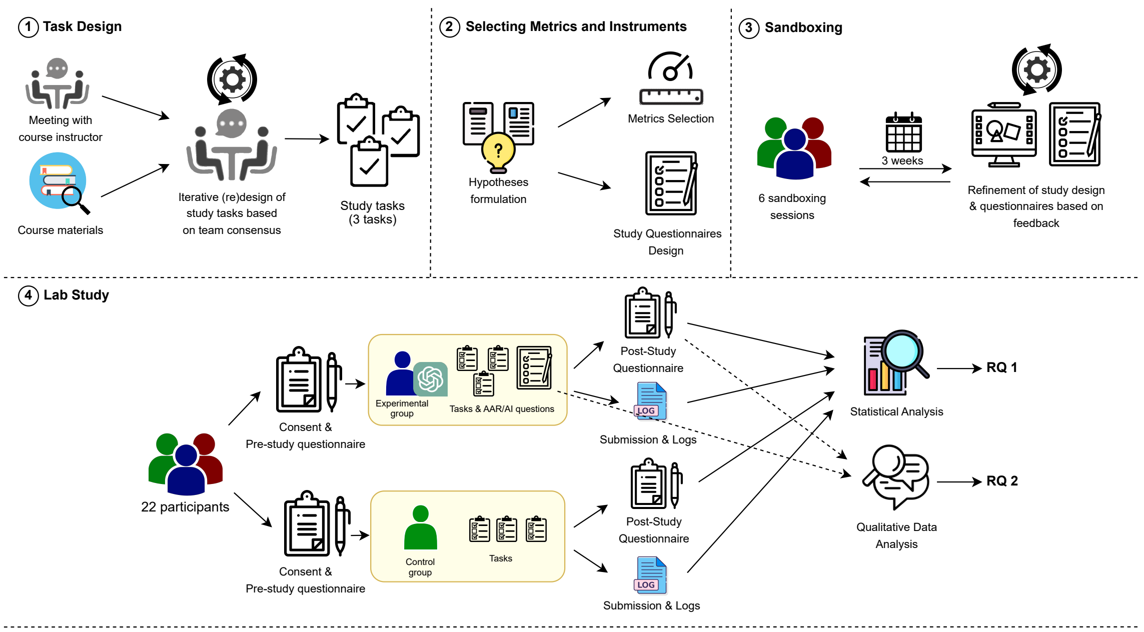

Figure 1: Overview of the research design

convo-genAI as it was state-of-the-art when writing this paper. We recruited 22 students enrolled in undergraduate software engineering courses at our university. Participants in both groups completed three SE tasks (fixed code functionalities, removed code smells, and contributed to a GitHub repository). We found no statistical differences between the two treatments in terms of participants' task performance or overall self-efficacy, but using ChatGPT increased participants' frustration.

Furthermore, to gain a comprehensive understanding of how convo-genAI can be effectively used and to inform its future design, it is essential to identify and evaluate its current pitfalls. This leads to our second research question: ***(RQ2): What are the current pitfalls in convo-genAI?*** Specifically, we investigated the faults that ChatGPT currently makes in the context of helping students in SE tasks, the causes of these faults, and their consequences for the participants. To answer this question, we employed After-Action Review for AI (AAR/AI) [33], a recent AI assessment process that allows end users to gauge AI faults [49] through a set of questions. The purpose of using AAR/AI was to assess the faults that participants in the Experimental group could perceive during their interaction with ChatGPT and its consequences for the participants. Qualitative analysis of these responses revealed *five fault categories and seven consequences stemming from these faults.*

Additionally, we used Microsoft's design guidelines for Human-AI Interaction (HAI) [3] as a lens to assess the ties between these faults and the violation of specific guidelines. Since these guidelines impact how users interact with the tool [53], it is important to get their perceptions of its violations. Doing so adds the benefit of participants reflecting on the tool and their interactions, thereby promoting metacognition [64]. A majority of participants reported that ChatGPT: (1) did not clearly outline its capabilities and limitations, (2) did not support efficient correction, (3) did not scope itself when in doubt, and (4) lacked transparency in its decision-making process. Our analysis revealed that these guideline violations were the root cause of the faults.

The primary contribution of this paper is an evaluation of the potential benefits and challenges of providing a convo-genAI system, specifically ChatGPT, to assist in software engineering tasks. Convo-genAI's ability to generate contextualized help along with natural language dialog capabilities can revolutionize software engineering. However, the inconsistency in its behavior and output pose significant challenges to its use in educational settings. Such systems, in their current state, can lead to user frustration and cognitive overload, which can be discouraging especially for novices, undermining their self-efficacy and potentially triggering an early exit from the field. Future design should thus incorporate the insights related to the current pitfalls and guideline violations to design for better user interactions. Educators, students, and self-learning professionals should also be aware of the consequences of using these tools to acquire new knowledge and skills.

2 METHOD

Our study goal is to understand the effectiveness and limitations of Conversational Generative AI (convo-genAI) compared to traditional online resources in helping software engineering students. To achieve this goal, we conducted a controlled experiment with students enrolled in undergraduate-level software engineering courses. Our study employed a between-subjects design, dividing the participants into two groups: an Experimental group that exclusively used ChatGPT (GPT-4); and a Control group that could use any online resources except genAI tools. We designed tasks based on in-depth discussions with the course instructor and careful analyses of the course materials. We measured multiple outcomes to comprehensively understand the effects and limitations of convo-genAI. The study protocol was refined through sandboxing sessions. After that, we conducted the experiment, as depicted in Figure 1.

2.1 Task Design

To design appropriate tasks, we first investigated students' current backgrounds and the skills they would learn in the course. Three researchers analyzed the course documentation and had in-depth discussions with the instructor about the learning objectives of the course. From the discussion, we learned that the students had low to medium familiarity with git and GitHub and that Python was the programming language used in the course. Furthermore, it was understood that the students were novices in software engineering and had a rudimentary understanding of good programming styles, API usage, and web scraping. We also learned that the students had not yet been formally introduced to the concept of code smells [38]. We tailored our tasks to align with the topics the instructor emphasized as key learning objectives in the course. Specifically, we focused on four topics: programmatic API usage, debugging, code quality, and version management, through the following 3 tasks: (1) fixing code functionalities involving third-party APIs, (2) removing code smells, and (3) contributing changes to a repository via a pull request.

The tasks were designed to be moderately challenging for the students. The research team held multiple meetings over two weeks to define and review the tasks. We then created a small Python program and hosted it in a GitHub repository[1]. We intentionally used a small program to manage task complexity. Additionally, we disorganized the code structure, added poorly crafted comments, and used non-intuitive function names to simulate common software engineering challenges. We conducted multiple sandboxing sessions with participants who had low to medium familiarity with software engineering, Git, and GitHub to validate the appropriateness and the level of complexity of these tasks:

Task 1: Participants needed to *create their own git branch* and *debug* code functionalities in their branch (3 subtasks). They were restricted from modifying some parts of the program or changing the test instances, ensuring that the only way to pass the validation was by fixing some code functionalities, as follows: (1) Participants had to correct the 'check_palindrome' function (logical programming exercise) that checked whether a number was a palindrome. We reversed the logic of this function, creating a bug. (2) Participants were asked to fix the 'check_weather' function (API usage exercise). The objective of the function was to retrieve weather data using the NOAA_SDK library[2] in Python, given specific coordinates. The correct method for this operation is to utilize the 'points_forecast' method from the API. We introduced a bug by using the 'get_forecasts' method, which actually takes the postal code and country as parameters, and passed latitude and longitude values instead. (3) Participants were tasked with fixing a function named 'check_webpage' (basic web scraping). The purpose of this function was to leverage the Selenium library[3] in Python to capture all visible elements from a webpage.

Task 2: The second task focused on improving *code quality by identifying and removing code smells*. We introduced these code smells: global variables, unused imports, dead code (commented-out code), and magic numbers (constants without context).

Task 3: The third task was designed to familiarize students with the *configuration management process* by contributing to a repository using git/GitHub. Participants were required to commit their contributions to the remote branch and create a pull request (PR) to the base branch. To maintain the integrity of the main branch, we instituted branch protection rules to prevent modifications in the base branch.

[1]See supplemental [5]: https://zenodo.org/record/8193821
[2]https://pypi.org/project/noaa-sdk/
[3]https://pypi.org/project/selenium/

2.2 RQs, Metrics and Instruments

We employed multiple metrics and instruments (Table 1) to understand students' perceptions of ChatGPT and the effects of interacting with it. In the following sections, we detail each RQ and the instruments used to answer them.

Table 1: Metrics and instruments and their relation to the RQs

Evaluation	Construct	Metrics, instruments, procedures, and literature
Students' perceptions of ChatGPT	Cognitive Load (RQ1)	NASA TLX [18, 45, 79]
	Perceived faults (RQ2)	AAR/AI [33, 49], Human-AI interaction guidelines [3]
	Continuance Intention (RQ1)	Post-study questionnaire
Effects of interacting with ChatGPT	Productivity (RQ1)	Task performance - Correctness [97], Time to complete [97]
	Consequences of AI faults (RQ2)	AAR/AI [33, 49]
	Self-efficacy (RQ1)	Self-efficacy questionnaire [82]

2.2.1 *RQ1: How effective is convo-genAI in helping students in software engineering tasks?*

Toward answering RQ1, we came up with three hypotheses and assessed the participants' intention to continue using ChatGPT, as detailed below.

H1: Students using ChatGPT for the study tasks perceive lower cognitive load than students using alternate resources.

Cognitive load is an important factor in designing effective scaffolding tools as it influences effective learning and end-user productivity [1]. Previous literature has shown that interacting with conversational agents (chatbots) significantly reduces cognitive load for end users in certain contexts [1, 18, 79]. Therefore, as a first step, we evaluated convo-genAI's impact on participants' perception of cognitive load in our context. *We hypothesized that students using ChatGPT for the study tasks (Experimental group) perceive a lower cognitive load compared to students who use alternate resources (Control group).* We used the original NASA Task Load Index (TLX)[45]—a validated and widely used post-study instrument—for measuring cognitive load across six dimensions (mental, physical, and temporal demand, performance, effort, and frustration).

H2: ChatGPT positively impacts students' productivity.

Schmidhuber et al. [79] revealed that chatbots can positively impact end users' productivity (measured in average correctness and average time required). Therefore, in our context, *we hypothesized that ChatGPT positively impacts students' productivity* and employed the variables from Xu et al.'s study [97] that assessed productivity in terms of task performance: (a) task correctness and (b) time to completion. However, we excluded (b) since we time-boxed the tasks (see Section 2.3) and participants in both groups utilized all of the allotted time for each task. To analyze the task correctness, two researchers used rubrics for each task—which is a prevalent approach in computer science education research [20, 24]. The rubrics were collaboratively developed beforehand (and are detailed in the supplemental [5]). The assessment consisted of number of tests correctly passed, code smells removed, successfully committed contributions and pull requests opened (assessed based on participants' code submissions and their GitHub log data). We

Table 2: AAR/AI steps and our adaptations. The Empirical context column explains how we realized the method in our study. Steps 3 to 6 were "inner loop" questions we repeated for all three tasks.

AAR/AI Steps	AAR/AI in our Empirical context
1. Defining the rules: How are we going to do this evaluation? What are the details regarding the situation?	We briefed the participants about the study details and how we were going to do the evaluation. Then we stated: "You will be given a questionnaire before and after each task. Please be detailed in your responses as that will help us evaluate ChatGPT's performance."
2. Explaining the objectives of the AI agent: What is the AI's objective for this situation?	We oriented the participants about the primary objective of ChatGPT by stating, "The primary objective of ChatGPT will be to assist you by providing contextual, disambiguous, and correct information."
Inner Loop	
3. Reviewing what was supposed to happen: What did the evaluator intend to happen?	We asked "What do you think should happen when you use ChatGPT for this task?" The participants chose between: It will (provide (all/some))/(not provide any) useful information I need to complete the task.
4. Identify what happened: What actually happened?	The participants did a task, then we asked "What actually happened when you used ChatGPT for this task?" The participants chose between: It (provided (all/some))/(did not provide any) useful information I need to complete the task.
5. Examine why it happened: Why did things happen the way they did?	We asked "Why do you think ChatGPT behaved this way?"
6. Formalize learning (end inner loop): What changes would you make in the decisions made by the AI to improve it?	We asked two questions: "To what extent did you modify ChatGPT's responses for solving the task?" The participants chose between: Did not modify at all/Modified (slightly/significantly). Then, we asked them to "Briefly explain why?"
End Inner Loop	
7. Formalize learning: What went well, what did not go well, what could be done differently next time?	We asked three questions: "What went well?", "What did not go well?", "What could be done differently next time?"

followed blind grading for code solutions (not knowing whether it came from the Experimental/Control group) to reduce potential bias.

H3: ChatGPT promotes students' self-efficacy.

Self-efficacy reflects an individual's perceived ability to successfully accomplish a task and can influence one's actual ability to complete a task [8]. Self-efficacy is considered a robust predictor of achievement [9] and motivation [25]. Recent studies have shown that conversational tools effectively promote students' self-efficacy [26, 68]. Hence, in our context, *we hypothesized that ChatGPT promotes students' self-efficacy.* To capture this construct, we adapted a questionnaire from Steinmacher et al. [82] and aligned it with the context of our study (see supplemental [5]). We used a 5-point Likert scale ('Strongly Disagree' to 'Strongly Agree') and administered the questionnaire in a before-after design, which allowed us to assess the impact of the resources on participants' self-perceived efficacy.

Users' continuance intention: Prior research highlights that the users' continuance intention reflects their satisfaction and positive perception towards the tool [7]. Park and Chung [69] evaluated continuance intention using a direct likelihood question. Echoing this method, we assessed the participants' continuance intention by presenting them with three statements: (1) Based on this experience, I plan to use ChatGPT to learn Software Engineering concepts; (2) Based on this experience, I do not plan to use ChatGPT to solve similar kinds of tasks; and (3) I would recommend ChatGPT to my friends if they need assistance with Software Engineering. The participants were asked to rate their level of agreement for each of these statements on a 5-point Likert scale ('Strongly Disagree' to 'Strongly Agree').

2.2.2 RQ2: What are the current pitfalls in convo-genAI?. To answer RQ2, we wanted participants from the Experimental group to assess AI faults. A consistent evaluation of an AI tool requires a standardized assessment process to avoid users adopting ad-hoc approaches, which can result in variations when evaluating the same AI tool. Therefore, we employed the After-Action Review for AI (AAR/AI) instrument [33], a standardized AI assessment process designed to aid humans in effectively gauging AI faults [49]. Khanna et al. [49] provided empirical evidence that integrating AAR/AI can aid end users in uncovering a significantly larger number of faults with greater precision. AAR/AI is a recent member of the After-Action Review (AAR) [63] family, devised by the U.S. military in the 1970s as a facilitated debriefing method. AAR has been used for decades and has been successfully adapted to different domains [46, 78].

The AAR/AI steps are derived from the "DEBRIEF" adaption by Sawyer and Deering [78]. There are 7 steps: **D**efining the rules, **E**xplaining the objectives of the AI agent, **B**enchmarking the performance of the agent, **R**eviewing what was supposed to happen, **I**dentifying what actually happened, **E**xamining why, and finally **F**ormalizing learning. AAR/AI is highly adaptable as it offers flexibility within each step of its process, accommodating customization to suit the specific needs of AI assessment in different domains. We adapted the AAR/AI steps as follows (Table 2):

Defining the rules and explaining the objectives: Each session started with the researcher providing an overview of the interfaces that the participants would use (Git/GitHub, Visual Studio Code) and the study tasks. We then told them the purpose of the assessment: ChatGPT was expected to deliver contextual, unambiguous, and accurate information (Steps 1-2, Table 2).

The inner loop: What & Why: Following task explanations and before each task initiation, participants answered, "What do you think should happen when you use ChatGPT for this task?" After each task, they responded to: "What actually happened when you used ChatGPT for this task?", "Why do you think ChatGPT behaved this way?" and "To what extent did you modify ChatGPT's responses for solving the task? Briefly explain why." (Steps 3-6, Table 2). The inner loop was repeated for all three tasks. The time to answer these questions was not counted towards task completion.

Formalizing learning: Once all tasks were completed, we asked the participants: "What went well?", "What did not go well?", and "What could be done differently next time?" (Step 7, Table 2).

To inform future design, assessing why the faults occur is important. Human-AI interaction guidelines inform AI system design and operation and can be used to understand where AI systems fail. Wright et al.'s [95] exploration of guidelines from three large tech companies (Apple, Google, and Microsoft) offered over 200 guidelines, ranging from initial considerations of AI to the curation of the models, the deployment of the AI-powered system, and the human-AI interface. Notably, Wright et al. found that while both Apple's [6] and Google's [43] guidelines were created with the developer in mind, Microsoft's guidelines [3] were designed with a keen focus on the user.

Therefore, we used Microsoft's guidelines [3] as our lens to examine whether ChatGPT's faults stemmed from potential guideline violations (the causes). Following this, we analyzed the effect of these faults on the participants (the consequences). We adapted the general recommendations to our context, and participants rated a set of statements using a 5-point Likert scale ('Strongly Disagree' to 'Strongly Agree'). These adaptations were made after team discussions and were refined during our sandboxing sessions.

2.3 Sandboxing

We conducted 6 sandbox sessions with CS students. We conducted the first two sessions via Zoom, and it became apparent that this setup posed several challenges (long setup times, limitations in our ability to control the environment, and difficulties in library installations). We decided to transition to an in-person lab setting. The sessions were planned to take 2 hours, but due to participant fatigue, we adjusted them to last 80 minutes and time-boxed the tasks: Task 1 was allotted 20 minutes, and tasks 2 and 3 had 10 minutes each. This change facilitated on-time conclusions and time for briefing and questionnaires (40 minutes).

Our sandboxing also revealed difficulties with the AAR/AI process: participants found it burdensome, resulting in sparse responses. Echoing a similar concern raised by Dodge et al. [33], we revised steps 3, 4, and 6—employing a mix of open and closed-ended questions—and adjusted the wording of the questions to improve their clarity. Additionally, we reverse-coded some items with negative connotations (low scores indicating agreement) in the Human-AI interaction guideline statements to counter acquiescence bias [10], a tendency where participants agree with statements regardless of their content or due to laziness, indifference, or automatic accommodation to a response pattern. Lastly, all the researchers agreed to omit Guideline 3 (time services based on context) due to its irrelevance in our context (ChatGPT remains idle unless prompted). Guideline 18 (notify users about changes) was also dropped, given its lack of applicability, as there were no major ChatGPT updates during the study's timeline (May 15 - June 2, 2023).

2.4 Lab Study

We conducted the lab studies over a period of three weeks.

Recruitment: We recruited undergraduate CS students who were at least 18 years of age and enrolled in software engineering courses at the university. We visited the classes in person and briefed them about the study. We asked those participants interested in the study to answer a survey about their demographic information (age, gender, academic level), resources they used for software engineering (GitHub, ChatGPT), and their experience in programming and software engineering. A total of 41 people responded to the survey.

Participants: After filtering out the incomplete responses, we invited 39 people, asking about their availability. We received 30 responses, but only 24 participants showed up for the study. We later discarded the data from 2 participants as their files were corrupted due to a fault in the machines. Out of the final 22 participants, 13 self-identified as men and 9 as women. We randomly assigned each participant to one of the two groups while allowing an even distribution of demographics and number of participants in each group. Participant IDs were assigned with the format 'PT-X' or 'PC-X' for the Experimental and Control groups, respectively. The participants' demographics are summarized in Table 3. As a token of appreciation, students received a $20 Amazon gift card.

Table 3: Participant demographics.

	Experimental	Control	Total
Man	6	7	13
Woman	5	4	9
Sophomore	2	1	3
Junior	5	5	10
Senior	4	5	9
Enrolled in SE Courses	3	4	7
Enrolled in SE Courses + Worked on SE Projects	8	7	15
Total	11	11	22

Study Protocol: All studies were in lab sessions at the University lab with up to 3 participants at a time, following the university IRB protocol. The study proceeded as follows: the participants agreed to an IRB-approved informed consent and were briefed about the different stages of the study, then filled out the pre-study questionnaire, performed the three tasks in the study (the Experimental group was also asked to fill out the AAR/AI questions before and after each task), and finally filled out the post-study questionnaire. The sessions were recorded with participants' consent and lasted around 80 minutes each. Before and after each study session, the browser histories and git branches were deleted to prevent unwarranted advantages and ensure all participants could start the session with the same information.

3 RESULTS

In the following, we present our findings of participant experiences with and without ChatGPT in the context of performing SE tasks.

3.1 RQ1: Effectiveness

To address RQ1 (effectiveness of students using ChatGPT for SE tasks), we tested our hypotheses (Section 2.2) using the Mann-Whitney U test [60].[4] The results are shown on Table 4.

To assess H1 (Cognitive Load), we examined the answers to TLX questions. As shown in Table 4, **frustration levels among participants using ChatGPT were significantly higher** than among those from the Control group (U=101, p=0.008***), with a large effect size (δ=0.669). Previous studies in Human-Robot Interaction highlight that end-user frustration is often induced by erroneous behavior in automated systems [91]. Our study corroborated similar

[4]We performed Shapiro-Wilk's test [80] for normality and Levene's test [19] for equal variances to determine the suitability of parametric tests. Since not all metrics met the assumptions for parametric tests, we used non-parametric tests for all hypotheses for consistency.

Table 4: Statistical results for cognitive load (NASA TLX) and task productivity (correctness).

	NASA TLX						Task Correctness			
	Mental	Physical	Temporal	Performance	Effort	**Frustration**	Task 1	Task 2	Task 3	Overall
Estimate	47	51.5	64.5	45.5	45.5	101	40	86	71.5	71
p-value	0.388	0.557	0.817	0.339	0.337	**0.008*****	0.16	0.078	0.303	0.507
Cliff's delta(δ)	-0.223	-0.149	0.066	-0.248	-0.248	**0.669**	**-0.339**	**0.421**	0.182	0.174
Median values for each group										
Experimental	15	1	15	9	14	14	1	1	2	4
Control	14	3	15	12	14	9	2	0	2	4

The estimates, p-values, and Cliff's delta (effect size) are with respect to Mann Whitney U-test. The **highlighted** columns are statistically significant. Negative δ suggests that the variable tends to have higher values in the Control group. We consider $|\delta| < [0, 0.15)$ to be no effect, $|\delta| \in [0.15, 0.33)$ to be small, $|\delta| \in [0.33, 0.47)$ to be medium, and $|\delta| > 0.47$ to be large, by convention [75].

patterns, highlighting an association between heightened frustration levels and faults in ChatGPT's behavior and responses. Several study participants clearly illustrated this. PT-3 conveyed, *"[ChatGPT] was fighting me a lot about the whole NOAA thing [Task 1, test case 2]."* Similar challenges were echoed by PT-7, who mentioned that *"[ChatGPT] misinterpreted my questions, was REALLY slow, and didn't account for errors."* Meanwhile, PT-1 articulated mistrust, *"I could not rely on [ChatGPT] to tell me when functions exist or not"*. Although the other factors were not statistically significant (and had small or negligible effect sizes), the participants using ChatGPT reported a slightly higher Mental demand (Med=15) compared to the Control group (Med=14) and perceived lower levels of Performance (Med=9) compared to others (Med=12). Still, Physical demand was rated very low for both groups, and there was no difference in Temporal nor Effort dimensions between the groups. Overall, *H1 is not supported*: the Experimental group had no significant advantages over the Control group across the cognitive load dimensions.

With respect to H2 (productivity), we could not find statistical differences in overall productivity (in terms of task correctness) between the two groups (Table 4). We observed a medium effect size pointing that participants using ChatGPT had lower productivity in terms of fixing code functionalities (Task 1: δ=-0.339) and higher productivity in terms of removing code smells (Task 2: δ=0.421). From these findings, it can be noted that although there were some variations in task-specific productivity, **we could not find an impact on productivity for the participants using ChatGPT** *(H2 is not supported).*

To assess how the allotted resources influenced participants' self-efficacy (H3), we analyzed the pre- and post-study response variations. The Wilcoxon-signed rank test did not show a statistically significant difference comparing the total self-efficacy score before and after ($p(Experimental) = 0.214$, $p(Control) = 0.7$).

Figure 2 shows the distribution of self-efficacy scores per question. We only noticed differences in terms of distributions and median shifts (comparing before/after for the groups) for Q2 and Q4. Overall, we observed a decrease in the Q2 scores (related to understanding Python code) for those who used ChatGPT, while the distribution remained the same for those who did not use it. For Q4 (related to the removal of code smells), the score for the group using ChatGPT increased after the task, while the median for the Control group remained the same (with a noticeable shift of the distribution towards negative scores). These trends align with the task-specific productivity findings (H2) observed with ChatGPT. For the other items, there were no differences when comparing the groups. We observed similar distributions and median shifts for the git/GitHub items (Q1, Q5, Q6) as well as for Q3 (fixing code functionalities)—comparing before and after for Experimental and Control groups. In summary, although ChatGPT influenced students' self-efficacy for certain items, overall, it **did not positively influence students' self-efficacy** (*H3 is not supported*).

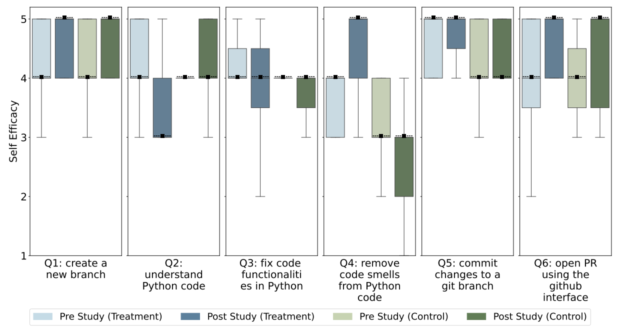

Figure 2: Self-efficacy results (box plots) per question. Medians are highlighted using black dots.

Furthermore, participants' continued intention to use ChatGPT (part of the post-study questionnaire—Table 1) received mixed responses (Figure 3). One respondent was undecided throughout (9.1%). The remaining participants were equally divided in their opinion about using ChatGPT and recommending it to friends needing assistance with software engineering. Five participants (45.5%) showed positive responses, and the other 5 (45.5%) provided negative responses, indicating a polarized view among students. Despite a segment expressing readiness to leverage the tool, an equal fraction expressed notable resistance: *"I would have liked to be able to ask someone knowledgeable in Python about [task 1] (PT-11)"*.

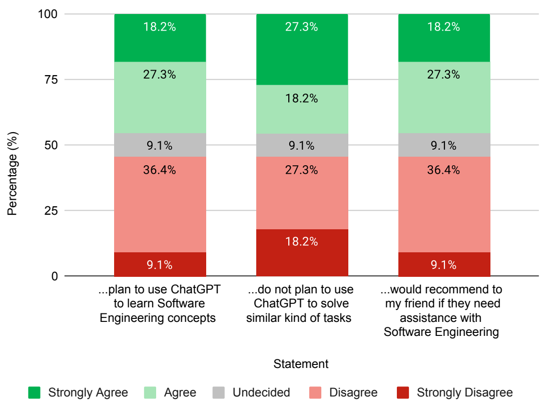

Figure 3: Continuance intention towards using ChatGPT (%)

3.2 RQ2: Pitfalls

This section presents the results for RQ2, i.e., the faults made by ChatGPT, their causes, and consequences within the context of assisting students in SE. Three authors qualitatively analyzed (open coding) the AAR/AI responses and identified ChatGPT's faults and their consequences on the participants. The coding was done collaboratively, with the authors engaging in iterative discussions (over 2 weeks) to reach a consensus on the final codes.

3.2.1 Faults made by ChatGPT. The faults made by ChatGPT in the context of assisting students in SE were grouped into 5 categories: (F1) Limited advice on niche topics, (F2) Inability to comprehend the problem, (F3) Incomplete assistance, (F4) Hallucination, and (F5) Wrong guidance.

F1: Limited advice on niche topics [PT-1, 5, 9]. ChatGPT struggled to provide expert advice on topics specific to a niche (e.g., a domain, a library, or a concept). For instance, PT-1 explains: *"ChatGPT only was helpful with general info, like git commands or logic issues. It wasn't helpful with niche specifics, like discerning between functions to use in a Python library"*. According to our participants, *"for anything that wasn't super standard, ChatGPT struggled to easily give useful answers. (PT-1)"* and thus *"having it define or explain ambiguous concepts did not help much (PT-5)"*. Moreover, ChatGPT provided limited advice regarding Python code functionalities (PT-1, 5, 9). This could be a reason for the **observed decrease in Task-1 specific productivity** for participants using ChatGPT for the study tasks. PT-9 said, *"ChatGPT did not have as much knowledge about the NOAA python library, and confidently told me incorrect ways to 'fix' my code."*

F2: Inability to comprehend the problem [PT-1, 5-8, 10]. ChatGPT could not always understand the participants' goals and the problems they were facing. Participants revealed that *"it incorrectly identified nonproblems as problems and missed actual problems (PT-1)"* and did not *"know the exact thing you want it to do despite giving it context (PT-6)"*. Before starting Task-1, 6 out of 11 participants (54%) responded that ChatGPT would provide all the required information. However, after completing the task, all participants marked that it only provided some information. This mismatch in expectations significantly **increased the participants' frustration levels**. PT-7 emphasized, *"[ChatGPT] misinterpreted my questions at times, was REALLY slow, and did not account for errors in code it provided me"*.

F3: Incomplete assistance [PT-1-3, 6-9, 11]. ChatGPT sometimes provided incomplete and partially correct assistance even when it was able to grasp the problem. PT-11 pointed out, *"I did ask ChatGPT questions about completing the task, but it did not give me answers on how to solve the whole task"*. Additionally, participants discovered that *"[ChatGPT] knows some things and can help give you advice on those things, but it won't immediately give you the correct answer (PT-6)"*. They also pointed out that, *"Some code provided by ChatGPT was correct, while some were incorrect and required modifying (PT-9)"*. This AI behavior likely **affected participants' mental workload**. As noted by PT-11, *"I could not figure out how to fix the import problem, and ChatGPT's suggestions didn't work"*.

F4: Hallucination [PT-4, 9, 11]. ChatGPT tends to hallucinate, creating false answers when it does not know the correct solution. Participants pointed out multiple instances of this. PT-4 stated that when ChatGPT *"did not have access to the documentation for the packages... it hallucinated answers"* and that it *"made up parameters for functions that were unfamiliar"*. Similarly, PT-11 noted that *"it did hallucinate sometimes, said there was a way to use a function in the noaa-sdk that was not possible"*. Additionally, there were instances of *confirmation bias* (when the AI conforms to the users' statements/requests, regardless of the actual accuracy/feasibility). PT-4 highlighted that ChatGPT *"was biased towards a 'yes, there are code smells' response...When they don't exist, it hallucinates them."*

F5: Wrong guidance [PT-2-4, 7-11]. In addition to hallucinating, there were other instances where ChatGPT gave wrong guidance, or *"incorrect ways to fix [problems] (PT-9)"*. For example, when it could not comprehend the problem (F2) participant (PT-8) was facing, it gave a piece of incorrect advice: *"It couldn't figure out test case 3 and kept telling me to check my drivers...without realizing there were missing imports (PT-8)"*. ChatGPT often *"did not account for errors in [solutions] it provided (PT-7)"*. It also suggested *"incorrect ways to fix [problems] (PT-9)"* when it had limited knowledge on a topic (F1) and hence some of its solutions appeared *"evidently wrong or unnecessary (PT-10)"*.

In summary, we observed that Experimental group participants' mental load and frustration increased when ChatGPT was unable to comprehend the problem (F2) or provided incomplete assistance (F3). Additionally, they perceived equal effort as the Control group participants, frequently tasked with identifying and resolving errors when ChatGPT hallucinated (F4) or provided wrong guidance (F5).

3.2.2 Causes of these faults and their consequences. For each of these faults, we examined why they occurred and the consequences they had on the participants (Figure 5). As mentioned in Section 2.2, participants rated Human-AI (HAI) guideline statements specific to ChatGPT interactions at the end of the experiment, which was used to assess guideline violations. We manually triangulated the response scores with open-ended text responses and found no discrepancies.

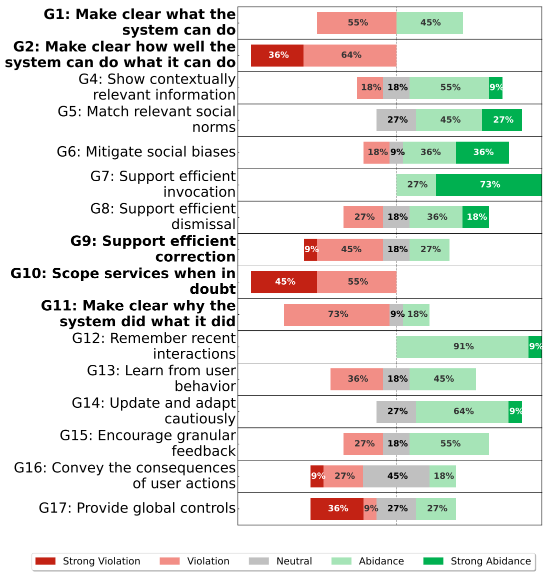

Figure 4: Human-AI Interaction guideline violations reported by participants; those found by more than 50% are in bold.

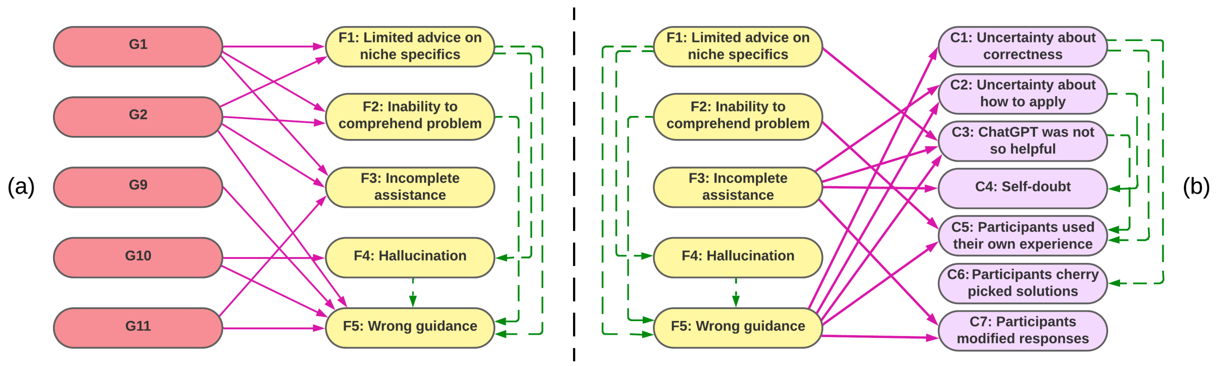
(a)
G1
G2
G9
G10
G11
F1: Limited advice on niche specifics
F2: Inability to comprehend problem
F3: Incomplete assistance
F4: Hallucination
F5: Wrong guidance
(b)
C1: Uncertainty about correctness
C2: Uncertainty about how to apply
C3: ChatGPT was not so helpful
C4: Self-doubt
C5: Participants used their own experience
C6: Participants cherry picked solutions
C7: Participants modified responses

Figure 5: The (a) causes and (b) consequences of ChatGPT's faults: Violation of Human-AI Interaction guidelines (G1, G2, ...) led to faults (F1, F2, ...). Faults had a cascading effect: one led to another and further led to consequences (C1, C2, ...) for participants. Some of these consequences led to other consequences.

Causes: All participants identified violations of Guideline 2 (G2: Make clear how well the system can do what it can do) and Guideline 10 (G10: Scope services when in doubt). Additionally, 8 out of 11 participants (~73%) found violations in Guideline 11 (G11: Make clear why the system did what it did) [PT-1-3, 5-7, 10-11], and 6 out of 11 participants (~54.5%) found violations in Guideline 1 (G1: Make clear what the system can do) [PT-1, 2, 4-6, 11] and Guideline 9 (G9: Support efficient correction) [PT-2-4, 7, 8, 11] (Figure 4). For each participant, we then mapped the faults (reported in AAR/AI) to the guideline violations they reported.

ChatGPT was perceived as offering limited advice on niche topics (F1), being unable to comprehend problems (F2), and providing incomplete assistance (F3), as no appropriate expectation of quality was set. Microsoft's HAI Guidelines 1 and 2 focus on clarifying expectations to prevent mismatches between users and AI, as demonstrated in prior literature [53]. Thus, it is likely that these faults resulted from ChatGPT's initial shortcomings in clearly stating its capabilities (G1 violation) and inadequately indicating how often it might make mistakes in its responses (G2 violation). There were instances when ChatGPT did not support efficient correction (G9 violation), making it difficult to refine its responses when it was incorrect: *"It was fighting me a lot...(PT-3)"*. Furthermore, ChatGPT also provided ambiguous/wrong information without conveying its uncertainty (G10 violation) and made it hard to gain an explanation regarding its decision-making process (G11 violation), likely resulting in hallucination (F4) and wrong guidance (F5).

Cascading faults: We also found that these faults had a cascading effect, where one fault led to another (green arrows in Figure 5). For instance, when ChatGPT struggled with niche specifics (F1) or was unable to comprehend problem (F2), it hallucinated (F4) and provided wrong guidance (F5): *"ChatGPT did not have as much knowledge . . . and confidently told me incorrect ways to 'fix' my code. (PT-9)," "kept telling me to check my drivers. . . without it realizing there were missing imports. (PT-8)."*

Consequences: From the AAR/AI responses, we identified 7 consequences for participants that arise due to ChatGPT's faults and grouped them into 3 categories: *Uncertainty* (uncertainty about correctness, uncertainty about how to apply), *Reflections* (ChatGPT was not so helpful, self-doubt), *Actions* (participants used their own experience, cherry-picked solutions, and modified the responses).

Participants grappled with how to apply ChatGPT's responses because of *uncertainty* and questioned its correctness:

C1: Uncertainty about correctness [PT-1, 2, 6, 10]. Participants were uncertain about the correctness of the responses provided by ChatGPT. For example, PT-1 stated, *"I could not rely on it to tell me when functions exist or not"*. PT-2 expressed skepticism, stating, *"I don't think it provided fully correct data, so I am inclined to pick parts only"*. This was related to the wrong guidance (F5) provided by ChatGPT, as per PT-10: *"Some answers look[ed] evidently wrong or unnecessary. It is important to modify the code based on my experience."*

C2: Uncertainty about how to apply [PT-1-3]. Participants were confused about how to apply/use the information provided by ChatGPT. As PT-1 expressed, *"I didn't use ChatGPT's responses since I wasn't sure how to apply them"*. We noticed that this consequence stemmed from two faults: incomplete assistance (F3)—*"got confused over suggestions w/ the weather library. . . correction that wasn't made obvious to me for palindrome (PT-2)"*—and wrong guidance (F5)—*"I'm not super sure why this didn't work. It was fighting me a lot about the whole NOAA thing (PT-3)."*

Participants' *reflection* (C3, C4) revealed that they found ChatGPT not so helpful and occasionally doubted themselves:

C3: ChatGPT was not so helpful [PT-1, 5, 10, 11]. We found instances where participants thought that ChatGPT was not so helpful. This was because ChatGPT provided limited advice on niche topics (F1), with PT-5 noting *"having [ChatGPT] define or explain ambiguous concepts did not help much"*. Other reasons included it providing incomplete assistance (F3), *". . . ChatGPT's suggestions didn't work (PT-11)"* and wrong guidance (F5): *"The suggestions...are not so helpful. It clearly gave wrong guidance (PT-10)"*.

C4: Self-doubt [PT-2, 4]. Participants doubted themselves, suspecting that they might be at fault. For instance, PT-2 shared, *"I got confused over suggestions with the weather library, likely I should have provided the full error... And I also may have asked something wrong? correction wasn't made obvious to me (PT-2)"*. This surfaced when ChatGPT delivered incomplete assistance (F3), leaving participants **uncertain about how to apply** the suggested solutions.

Participants reported that they had to undertake *actions* (C5, C6, C7) to tackle portions of tasks on their own (no reliance), cherry-pick from provided solutions and modify responses provided by ChatGPT (partial reliance):

C5: Participants used their own experience [PT-1, 4, 7, 10].

Participants had to use their own experience to tackle certain aspects of the tasks, particularly when ChatGPT was unable to comprehend their problem (F2) and thus was perceived to be **not so helpful**: *"Everything else I had to do on my own because ChatGPT didn't comprehend enough to help (PT-1)"*. Participants devised their own solutions when they were **uncertain of the correctness** of ChatGPT's suggestions, attributed to its tendency to provide wrong guidance (F5): *"Some answers look[ed] evidently wrong or unnecessary. It is important to modify the code based on my experience (PT-10)"*.

C6: Participants cherry-picked solutions [PT-2, 3, 5]. Participants picked parts of solutions provided by ChatGPT that seemed correct: *"Some of the things I didn't necessarily agree w/ but some of it was valid, so I picked & chose what I liked (PT-3)"*. When **uncertain about correctness**, participants were *"inclined to pick parts only (PT-2)"*.

C7: Participants modified the responses [PT-6, 7, 9, 10]. Participants modified the responses provided by ChatGPT to come up with the correct solution. PT-7 noted, *"I used its responses in tandem so I kind of combined them. When it wasn't right I did it myself"*. This consequence stemmed from ChatGPT's incomplete assistance (F3) and also wrong guidance (F5): *"Some code provided by ChatGPT was correct, while some was incorrect and required modifying it (PT-9)"*.

4 DISCUSSION: RECOMMENDATION

It is necessary to customize Generative AI as an effective scaffolding learning agent for software engineering. While genAI tools have proven effective in providing quick solutions to user queries, this approach conflicts with the traditional goals of education. Directly giving away answers can diminish the need for critical thinking and impact learning, potentially leading to reduced self-efficacy [30] and motivation [25]. In our study, we observed that participants' self-efficacy decreased in certain cases, like understanding Python code (see Sect. 3.1, Fig. 2). This decline may be attributed to students viewing these tools as advanced search engines that offer ready-made solutions, a practice that instructors fear could impede genuine learning [51].

Hence, future research should investigate how genAI can be tailored and optimized as an effective scaffolding learning agent. To be effective, a scaffolding agent must correctly interpret the students' intentions, an aspect where genAI shows promising results. Nonetheless, more work is necessary to understand the expectations of students and instructors and how students express their expectations and engage in dialogue with the agent. These insights can help design agents that grasp students' intentions and adapt their interactions to enhance pedagogical outcomes. Further research can also explore how genAI can be leveraged for personalized student assistance, using techniques like the 'persona prompt pattern' [93] to adjust content based on expertise levels. Moreover, future research should consider incorporating pedagogical scaffolds (templates, heuristics, or human intervention) into AI-generated content to clarify the AI's problem-solving process and handle underperforming scenarios.

Recommendations for future Generative AI design: Participants using ChatGPT in our study considered that it violated 5 out of the 18 HAI guidelines. These participants also perceived lower performance levels and uncertainty, which led to self-doubt and possibly lowered their self-efficacy: *"I got confused over suggestions...I may have asked something wrong. (PT-2)", "I'm not super sure why this didn't work (PT-3)"*. When participants were *uncertain*, they had to use their own experience (C5) in addition to cherry-picking solutions (C6) and modifying ChatGPT's responses (C7) to fit their context. These likely added to the participants' *mental workload* and possibly explain why we could not observe a positive impact on *task productivity*.

Prior literature highlights that implementing Microsoft's HAI Guidelines has substantiated effects on users, including *increased trust, decreased suspicion* along with them *feeling more in control, less inadequate, more productive, secure, and less uncertain* [53]. Based on this, we suggest that an iterative participatory approach [81] should be followed in the future design of genAI systems to ensure that the systems adhere to the HAI guidelines it currently violates: clearly stating capabilities (G1) and limitations (G2), supporting efficient correction (G9), scoping services when in doubt (G10), and maintaining transparency in the decision-making process (G11). Furthermore, Wang et al. [90] recently identified specific design strategies for genAI systems based on their design probe study. They found that communicating AI performance via usage statistics, offering indicators of model mechanisms to support evaluation, and allowing users to configure AI by adjusting preferences helped set proper expectations and inform appropriate usage of AI tools. We speculate that incorporating these design practices and strategies can also mitigate the observed negative consequences on students and significantly foster appropriate levels of trust [47] in genAI-based scaffolding tools.

Building Inclusive Technology: Like any other tool, AI systems can embed cognitive biases [88] arising from a lack of support for cognitive diversity [67]. In our study, we found considerable disparities in perceived violations of the HAI guidelines on disaggregating participants' data based on their gender. All women reported a violation of Guideline 11, perceiving ChatGPT's decision-making process as hard to explain, whereas only 3 out of 6 men reported this violation. This could be because women tend to favor comprehensive information-processing style [62] and ChatGPT lacked transparency in its decision-making process. Individuals with this information processing style tend to seek out all the information needed before starting a task, whereas those who are selective information processors take the first piece of actionable advice and work on it. Similarly, a majority of men marked that ChatGPT did not personalize their experience by learning from their actions over time (4 out of 6 men perceived Guideline 13 violation) and did not allow for global customization of its behavior (5 out of 6 men perceived Guideline 17 violation). In contrast, no women found these guidelines to be in violation. This disparity was likely because of the limited tinkering allowed around its interface and technology.

Gender HCI research [11, 76] has found that tools often lack support for diverse cognitive styles. As a result, individuals whose styles are not accommodated face *cognitive bias bugs*—an additional cognitive tax they pay when they use the tool. Further, individual differences in how people solve problems and use software cluster by gender [23]; i.e., some styles are favored more by men than women, and vice-versa. Research spanning a decade has identified that women are often more risk-averse than men [27] and prefer process-oriented learning and are thus less likely to tinker [12, 21].

This implies that if future AI tools are not inclusive of cognitive styles, they will not be inclusive of gender. Previous research has also reported that user interactions and experiences with AI systems are significantly diverse for diverse users [4]. Indeed, the current genAI systems have faced criticism for their potential negative impacts on equity [14, 55]. In future research, it would be valuable to explore how uncertainty, similar to what we identified in our study, may be associated with cognitive styles. Understanding these connections could pave the way for implementing specific strategies that effectively address and mitigate the impact of uncertainty in the context of AI-assisted learning. To achieve this, the GenderMag method [23] can be applied for evaluating AI systems throughout their iterative design cycles. Past work has shown that fixing issues found from using GenderMag-based processes creates more inclusive tools and environments [22, 23, 67, 76].

5 RELATED WORK

Chatbots have been popular in educational settings [50, 65, 94]. In software engineering education, chatbots have been proposed as a way to support students outside of the classroom [16]: offering expert advice in problem-solving activities [87] and accompanying students in their capstone projects [41]. However, these chatbots are often confined to queries anticipated by educators [34], thereby serving as a programmed search feature for frequent queries.

Generative AI models have demonstrated an impressive ability to solve a large number of computation problems from natural language prompts [28], and are now being used for programming tasks [54, 96]. Recent literature highlights how these models outperform most students on typical CS1 and CS2 exam problems [36, 37], handle variations in problem-wording [36], and even surpass human performance on programming competitions [54]. Researchers have further explored how these models can be used to enhance learning computer science. Sarsa et al. [77] used genAI to create coding exercises and explanations, both of which can be used to provide practice and guidance to students. Another study combined Codex with learnersourcing (crowdsourcing for learners) to create and validate exercises that are engaging for learners [32].

Generative models have also been used to aid instructors by automating content creation for interactive course materials [58, 59]. Further, Denny et al. [31] showed that prompt engineering is effective in improving AI responses and thus could give instructors more control to improve the relevancy of generated materials.

Another line of research evaluates the impact of genAI tools on students. Prather et al. [74] studied students' initial impressions of using Copilot for CS1 programming tasks. In a different study, Kazemitabaar et al. [48] conducted an experiment comparing pre-college students learning Python with and without Codex's help. Their findings indicated that Codex users improved their coding skills more than their counterparts, though both groups achieved similar conceptual comprehension. However, Bird et al. noted that while Codex (Copilot) expedited code writing, it compromised learners' code comprehension [17]. Moreover, Vaithilingam et al. [85] reported that despite the initial user interest in genAI, these tools did not enhance users' task efficiency (time) or accuracy.

Our work complements these, as it looks specifically into supporting students in software engineering.

6 THREATS TO VALIDITY

Construct Validity: We used instruments from the literature [3, 45, 49, 82, 97] to measure our constructs as much as possible since they had already been used and validated in other contexts. We assessed the instruments adapted to our context with sandbox sessions and refined the instruments and research protocol until the team was confident of the instruments' reliability. Nevertheless, despite our efforts, we acknowledge that questions might be misinterpreted and can lead to incorrect measurements.

Internal Validity: We acknowledge that our study, like others, can have self-selection bias, where participants interested in the topic of the study were motivated to participate. Participant exhaustion and distraction might have also affected the study results. We mitigated this threat by limiting the length of sessions (80 minutes) and time-boxing the tasks. However, this could mean that participants did not have enough time to complete the tasks. Since participant interaction was the primary focus and time-boxing was applied to both treatments, this does not impact the validity of between-group comparisons. Another potential threat is task selection, where study tasks can be too easy or too complicated for our target population. We mitigated this risk by designing the tasks based on the instructor's insights and course materials and sandboxing them with participants of varying expertise. Further, participants assessed the ChatGPT interactions—our investigation focus—to identify faults and guideline violations, which makes the findings dependent on their capabilities. We believe this is not a problem since participants had prior experience with ChatGPT and Python (an average of 3.75 years), and thus had the needed experience to assess their interactions with ChatGPT. Past works [33, 49] that employed similar instruments have recruited participants with at least 10 hours of experience, which puts us in line with them. Finally, desirability bias may have an impact because participants may have favored ChatGPT due to its hype. To mitigate this threat, we adapted neutral and non-judgmental language to frame the questions and explain the experiment and analyzed data cautiously, acknowledging the potential presence of desirability bias when interpreting the results.

Reliability: Interpreting qualitative data can be challenging and potentially affect the study's validity. To ensure consistency, we employed robust techniques from existing literature and continually compared our analysis with established codes. Additionally, we held frequent meetings to discuss and refine the codes and categories until we reached a unanimous agreement.

External Validity: We structured our tasks in Python, which was used in the software engineering courses and students were most familiar with it. Therefore, we trade off generalizability for depth in our context, and our findings might not necessarily generalize to other programming languages, universities, convo-genAI systems, and work contexts. Further, different interaction styles with ChatGPT can lead to different outcomes, and the participants in the study might not have represented all styles. The relatively small sample size of 22 participants is also a threat to the generalizability of the study. We mitigated this threat by recruiting participants from multiple software engineering classes and evenly distributing them in each group based on their demographics.

7 CONCLUSION

Our work comprehensively evaluates convo-genAI's potential and pitfalls in supporting software engineering tasks. Our analysis did not reveal any statistical differences in participants' productivity or self-efficacy in using ChatGPT for the study tasks compared to traditional resources. ChatGPT, in its current state, increased participants' frustration levels, led to uncertainty, and in some cases induced self-doubt, due to the lack of transparency and clarity in its behavior and communication. This highlights the need for caution; while it provides good answers in straightforward cases, it tends to give incorrect or confusing responses in more complex scenarios. *"For anything that wasn't super standard, ChatGPT struggled to easily give useful answers (PT-1)"*—Expert developers can navigate this, finding value in the AI responses, but novices might struggle or learn incorrect practices.

Our findings provide foundational insights for future convo-genAI design towards enhanced human-AI interaction, subsequently informing the current consequences of using such tools for acquiring new knowledge and skills. We foresee that with careful co-design, genAI holds immense potential in helping novices learn software engineering. We plan to use the insights from this study to implement and evaluate genAI-embedded pedagogical tools that foster critical thinking and support learning.

ACKNOWLEDGMENTS

We thank Samarendra Hedaoo for his insights in the meetings about the SE course. We thank all participants who took part in the study for their time and effort. We thank members of the EPIC lab at the university: Amreeta Chatterjee, Emily Garcia, Mariam Guizani, Zixuan Feng, and Bianca Trinkenreich, among others, for their support and valuable feedback, and Andrew Anderson for his early insights. This work was partially supported by the National Science Foundation under Grant Numbers: 2235601, 2236198, 2247929, 2303042, and 2303043. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

REFERENCES

[1] Suhni Abbasi, Hameedullah Kazi, and Nazish Nawaz Hussaini. 2019. Effect of chatbot systems on students learning outcomes. *Sylwan* 163, 10 (2019).

[2] Larry Alton. 2017. Phone calls, texts or email? Here's how millennials prefer to communicate. *Forbes. com. Available at: https://www. forbes. com/sites/larryalton/2017/05/11/how-do-millennials-prefer-to-communicate* (2017).

[3] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N Bennett, Kori Inkpen, et al. 2019. Guidelines for human-AI interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–13.

[4] Andrew Anderson, Tianyi Li, Mihaela Vorvoreanu, and Margaret Burnett. 2021. Diverse Humans and Human-AI Interaction: What Cognitive Style Disaggregation Reveals. *arXiv preprint arXiv:2108.00588* (2021).

[5] Anonymous. 2023. Supplemental Material for ChatGPT User Study . https://doi.org/10.5281/zenodo.8193821

[6] Apple. 2023. Human interface guidelines for machine learning. https://developer.apple.com/design/human-interface-guidelines/machine-learning/.

[7] Muhammad Ashfaq, Jiang Yun, Shubin Yu, and Sandra Maria Correia Loureiro. 2020. I, Chatbot: Modeling the determinants of users' satisfaction and continuance intention of AI-powered service agents. *Telematics and Informatics* 54 (2020), 101473.

[8] Albert Bandura. 1986. The explanatory and predictive scope of self-efficacy theory. *Journal of Social and Clinical Psychology* 4, 3 (1986), 359–373.

[9] Albert Bandura. 1993. Perceived self-efficacy in cognitive development and functioning. *Educational Psychologist* 28, 2 (1993), 117–148.

[10] Hans Baumgartner and Jan-Benedict EM Steenkamp. 2001. Response styles in marketing research: A cross-national investigation. *Journal of Marketing Research* 38, 2 (2001), 143–156.

[11] Laura Beckwith, Margaret Burnett, Valentina Grigoreanu, and Susan Wiedenbeck. 2006. Gender HCI: What about the software? *Computer* 39, 11 (2006), 97–101.

[12] Laura Beckwith, Cory Kissinger, Margaret Burnett, Susan Wiedenbeck, Joseph Lawrance, Alan Blackwell, and Curtis Cook. 2006. Tinkering and gender in end-user programmers' debugging. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 231–240.

[13] Andrew Begel and Beth Simon. 2008. Novice software developers, all over again. In *Proceedings of the Fourth International Workshop on Computing Education Research*. 3–14.

[14] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big?. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*. 610–623.

[15] Patrick Bii. 2013. Chatbot technology: A possible means of unlocking student potential to learn how to learn. *Educational Research* 4, 2 (2013), 218–221.

[16] Mikas Binkis, Ramūnas Kubiliūnas, Rima Sturienė, Tatjana Dulinskienė, Tomas Blažauskas, and Vitalija Jakštienė. 2021. Rule-Based Chatbot Integration into Software Engineering Course. In *Information and Software Technologies: 27th International Conference, ICIST 2021, Kaunas, Lithuania, October 14–16, 2021, Proceedings 27*. Springer, 367–377.

[17] Christian Bird, Denae Ford, Thomas Zimmermann, Nicole Forsgren, Eirini Kalliamvakou, Travis Lowdermilk, and Idan Gazit. 2022. Taking Flight with Copilot: Early insights and opportunities of AI-powered pair-programming tools. *Queue* 20, 6 (2022), 35–57.

[18] Florian Brachten, Felix Brünker, Nicholas RJ Frick, Björn Ross, and Stefan Stieglitz. 2020. On the ability of virtual agents to decrease cognitive load: an experimental study. *Information Systems and e-Business Management* 18 (2020), 187–207.

[19] Morton B Brown and Alan B Forsythe. 1974. Robust tests for the equality of variances. *J. Amer. Statist. Assoc.* 69, 346 (1974), 364–367.

[20] Khalid Abdulmohsen Buragga, Abdul Raouf Khan, Noor Zaman, et al. 2013. Rubric based assessment plan implementation for Computer Science program: A practical approach. In *Proceedings of 2013 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*. IEEE, 551–555.

[21] Margaret Burnett, Scott D Fleming, Shamsi Iqbal, Gina Venolia, Vidya Rajaram, Umer Farooq, Valentina Grigoreanu, and Mary Czerwinski. 2010. Gender differences and programming environments: across programming populations. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. 1–10.

[22] Margaret Burnett, Anicia Peters, Charles Hill, and Noha Elarief. 2016. Finding Gender-Inclusiveness Software Issues with GenderMag: A Field Investigation. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, 2586–2598. https://doi.org/10.1145/2858036.2858274

[23] Margaret Burnett, Simone Stumpf, Jamie Macbeth, Stephann Makri, Laura Beckwith, Irwin Kwan, Anicia Peters, and William Jernigan. 2016. GenderMag: A Method for Evaluating Software's Gender Inclusiveness. *Interacting with Computers* 28, 6 (10 2016), 760–787. https://doi.org/10.1093/iwc/iwv046

[24] Veronica Cateté and Tiffany Barnes. 2017. Application of the Delphi method in computer science principles rubric creation. In *Proceedings of the 2017 ACM conference on innovation and technology in computer science education*. 164–169.

[25] Chiung-Sui Chang, Eric Zhi-Feng Liu, Hung-Yen Sung, Chun-Hung Lin, Nian-Shing Chen, and Shan-Shan Cheng. 2014. Effects of online college student's Internet self-efficacy on learning motivation and performance. *Innovations in Education and Teaching International* 51, 4 (2014), 366–377.

[26] Ching-Yi Chang, Gwo-Jen Hwang, and Meei-Ling Gau. 2022. Promoting students' learning achievement and self-efficacy: A mobile chatbot approach for nursing training. *British Journal of Educational Technology* 53, 1 (2022), 171–188.

[27] Gary Charness and Uri Gneezy. 2012. Strong evidence for gender differences in risk taking. *Journal of Economic Behavior & Organization* 83, 1 (2012), 50–58.

[28] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374* (2021).

[29] Marian Daun and Jennifer Brings. 2023. How ChatGPT Will Change Software Engineering Education. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*. 110–116.

[30] Marzieh Dehghani, Hamideh Pakmehr, Asma Malekzadeh, et al. 2011. Relationship between students' critical thinking and self-efficacy beliefs in Ferdowsi University of Mashhad, Iran. *Procedia-Social and Behavioral Sciences* 15 (2011), 2952–2955.

[31] Paul Denny, Viraj Kumar, and Nasser Giacaman. 2023. Conversing with copilot: Exploring prompt engineering for solving cs1 problems using natural language. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. 1136–1142.

[32] Paul Denny, Sami Sarsa, Arto Hellas, and Juho Leinonen. 2022. Robosourcing Educational Resources–Leveraging Large Language Models for Learnersourcing.

arXiv preprint arXiv:2211.04715 (2022).
[33] Jonathan Dodge, Roli Khanna, Jed Irvine, Kin-Ho Lam, Theresa Mai, Zhengxian Lin, Nicholas Kiddle, Evan Newman, Andrew Anderson, Sai Raja, et al. 2021. After-action review for AI (AAR/AI). *ACM Transactions on Interactive Intelligent Systems (TiiS)* 11, 3-4 (2021), 1–35.
[34] Juan Carlos Farah, Basile Spaenlehauer, Vandit Sharma, María Jesús Rodríguez-Triana, Sandy Ingram, and Denis Gillet. 2022. Impersonating chatbots in a code review exercise to teach software engineering best practices. In *2022 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 1634–1642.
[35] Umer Farooq and Jonathan Grudin. 2016. Human-computer integration. *Interactions* 23, 6 (2016), 26–32.
[36] James Finnie-Ansley, Paul Denny, Brett A Becker, Andrew Luxton-Reilly, and James Prather. 2022. The robots are coming: Exploring the implications of openai codex on introductory programming. In *Proceedings of the 24th Australasian Computing Education Conference*. 10–19.
[37] James Finnie-Ansley, Paul Denny, Andrew Luxton-Reilly, Eddie Antonio Santos, James Prather, and Brett A Becker. 2023. My AI Wants to Know if This Will Be on the Exam: Testing OpenAI's Codex on CS2 Programming Exercises. In *Proceedings of the 25th Australasian Computing Education Conference*. 97–104.
[38] Martin Fowler. 1997. Refactoring: Improving the design of existing code. In *11th European Conference. Jyväskylä, Finland*.
[39] Luke K Fryer, Mary Ainley, Andrew Thompson, Aaron Gibson, and Zelinda Sherlock. 2017. Stimulating and sustaining interest in a language course: An experimental comparison of Chatbot and Human task partners. *Computers in Human Behavior* 75 (2017), 461–468.
[40] Emily Arteaga Garcia, João Felipe Pimentel, Zixuan Feng, Marco Gerosa, Igor Steinmacher, and Anita Sarma. 2023. How to Support ML End-User Programmers through a Conversational Agent. In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE)*. IEEE Computer Society, 618–629.
[41] Luis A Gonzalez, Andres Neyem, Ignacio Contreras-McKay, and Danilo Molina. 2022. Improving learning experiences in software engineering capstone courses using artificial intelligence virtual assistants. *Computer Applications in Engineering Education* 30, 5 (2022), 1370–1389.
[42] Google. 2023. Bard. https://bard.google.com/.
[43] Google. 2023. People+ai guidebook. https://pair.withgoogle.com/guidebook/.
[44] Jonathan Grudin and Richard Jacques. 2019. Chatbots, humbots, and the quest for artificial general intelligence. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–11.
[45] Sandra G Hart and Lowell E Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In *Advances in psychology*. Vol. 52. Elsevier, 139–183.
[46] Andrew W Ishak and Elizabeth A Williams. 2017. Slides in the tray: How fire crews enable members to borrow experiences. *Small Group Research* 48, 3 (2017), 336–364.
[47] Brittany Johnson, Christian Bird, Denae Ford, Nicole Forsgren, and Thomas Zimmermann. 2023. Make Your Tools Sparkle with Trust: The PICSE Framework for Trust in Software Tools. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 409–419.
[48] Majeed Kazemitabaar, Justin Chow, Carl Ka To Ma, Barbara J Ericson, David Weintrop, and Tovi Grossman. 2023. Studying the effect of AI Code Generators on Supporting Novice Learners in Introductory Programming. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–23.
[49] Roli Khanna, Jonathan Dodge, Andrew Anderson, Rupika Dikkala, Jed Irvine, Zeyad Shureih, Kin-ho Lam, Caleb R Matthews, Zhengxian Lin, Minsuk Kahng, et al. 2022. Finding AI's faults with AAR/AI: An empirical study. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 12, 1 (2022), 1–33.
[50] Mohammad Amin Kuhail, Nazik Alturki, Salwa Alramlawi, and Kholood Alhejori. 2023. Interacting with educational chatbots: A systematic review. *Education and Information Technologies* 28, 1 (2023), 973–1018.
[51] Sam Lau and Philip J Guo. 2023. From" Ban It Till We Understand It" to" Resistance is Futile": How University Programming Instructors Plan to Adapt as More Students Use AI Code Generation and Explanation Tools such as ChatGPT and GitHub Copilot. (2023).
[52] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597* (2023).
[53] Tianyi Li, Mihaela Vorvoreanu, Derek DeBellis, and Saleema Amershi. 2022. Assessing Human-AI Interaction Early through Factorial Surveys: A Study on the Guidelines for Human-AI Interaction. *ACM Transactions on Computer-Human Interaction* (2022).
[54] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. 2022. Competition-level code generation with alphacode. *Science* 378, 6624 (2022), 1092–1097.
[55] Paul Pu Liang, Chiyu Wu, Louis-Philippe Morency, and Ruslan Salakhutdinov. 2021. Towards understanding and mitigating social biases in language models. In *International Conference on Machine Learning*. PMLR, 6565–6576.
[56] Dastyni Loksa, Amy J Ko, Will Jernigan, Alannah Oleson, Christopher J Mendez, and Margaret M Burnett. 2016. Programming, problem solving, and self-awareness: Effects of explicit guidance. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 1449–1461.
[57] Wei Ma, Shangqing Liu, Wenhan Wang, Qiang Hu, Ye Liu, Cen Zhang, Liming Nie, and Yang Liu. 2023. The Scope of ChatGPT in Software Engineering: A Thorough Investigation. *arXiv preprint arXiv:2305.12138* (2023).
[58] Stephen MacNeil, Andrew Tran, Arto Hellas, Joanne Kim, Sami Sarsa, Paul Denny, Seth Bernstein, and Juho Leinonen. 2023. Experiences from using code explanations generated by large language models in a web software development e-book. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. 931–937.
[59] Stephen MacNeil, Andrew Tran, Dan Mogil, Seth Bernstein, Erin Ross, and Ziheng Huang. 2022. Generating diverse code explanations using the gpt-3 large language model. In *Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 2*. 37–39.
[60] Patrick E McKnight and Julius Najab. 2010. Mann-Whitney U Test. *The Corsini encyclopedia of psychology* (2010), 1–1.
[61] Meta. 2023. Llama2. https://ai.meta.com/llama/.
[62] Joan Meyers-Levy and Barbara Loken. 2015. Revisiting gender differences: What we know and what lies ahead. *Journal of Consumer Psychology* 25, 1 (2015), 129–149.
[63] John E Morrison and Larry L Meliza. 1999. *Foundations of the after action review process*. Technical Report. Institute for Defense Analyses Alexandria Va.
[64] Thomas O Nelson and Louis Narens. 1994. Why investigate metacognition. *Metacognition: Knowing about knowing* 13 (1994), 1–25.
[65] Chinedu Wilfred Okonkwo and Abejide Ade-Ibijola. 2021. Chatbots applications in education: A systematic review. *Computers and Education: Artificial Intelligence* 2 (2021), 100033.
[66] OpenAI. 2023. GPT-4. https://openai.com/product/gpt-4.
[67] Hema Susmita Padala, Christopher Mendez, Felipe Fronchetti, Igor Steinmacher, Zoe Steine-Hanson, Claudia Hilderbrand, Amber Horvath, Charles Hill, Logan Simpson, Margaret Burnett, et al. 2020. How gender-biased tools shape newcomer experiences in oss projects. *IEEE Transactions on Software Engineering* 48, 1 (2020), 241–259.
[68] Doeun Park, Myounglee Choo, Bohyun Jin, Un Sun Chung, Jinwoo Kim, Junghan Lee, and Yee-Jin Shin. 2023. Utilizing a Conversational Agent to Promote Self-efficacy in Children: A Pilot Study on Low Cognitive Ability Children with Attention Deficit Hyperactivity Disorder. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–7.
[69] MY Park and KH Chung. 2011. The antecedents and consequences of user satisfaction in virtual community: Focused on college students. *Korean Research Academy of Distribution and Management Review* 14, 1 (2011), 77–99.
[70] Sida Peng, Eirini Kalliamvakou, Peter Cihon, and Mert Demirer. 2023. The impact of ai on developer productivity: Evidence from github copilot. *arXiv preprint arXiv:2302.06590* (2023).
[71] Juanan Pereira. 2016. Leveraging chatbots to improve self-guided learning through conversational quizzes. In *Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality*. 911–918.
[72] Gustavo Pinto, Clarice Ferreira, Cleice Souza, Igor Steinmacher, and Paulo Meirelles. 2019. Training software engineers using open-source software: the students' perspective. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. IEEE, 147–157.
[73] G. Pinto, F. Figueira Filho, I. Steinmacher, and M. A. Gerosa. 2017. Training Software Engineers Using Open-Source Software: The Professors' Perspective. In *The 30th IEEE Conference on Software Engineering Education and Training*. 1–5.
[74] James Prather, Brent N Reeves, Paul Denny, Brett A Becker, Juho Leinonen, Andrew Luxton-Reilly, Garrett Powell, James Finnie-Ansley, and Eddie Antonio Santos. 2023. " It's Weird That it Knows What I Want": Usability and Interactions with Copilot for Novice Programmers. *arXiv preprint arXiv:2304.02491* (2023).
[75] Jeanine Romano, Jeffrey D Kromrey, Jesse Coraggio, Jeff Skowronek, and Linda Devine. 2006. Exploring methods for evaluating group differences on the NSSE and other surveys: Are the t-test and Cohen'sd indices the most appropriate choices. In *Annual meeting of the Southern Association for Institutional Research*. Citeseer, 1–51.
[76] Ítalo Santos, João Felipe Pimentel, Igor Wiese, Igor Steinmacher, Anita Sarma, and Marco Aurélio Gerosa. 2023. Designing for Cognitive Diversity: Improving the GitHub Experience for Newcomers. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. IEEE, 12 pages.
[77] Sami Sarsa, Paul Denny, Arto Hellas, and Juho Leinonen. 2022. Automatic generation of programming exercises and code explanations using large language models. In *Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 1*. 27–43.
[78] Taylor Lee Sawyer and Shad Deering. 2013. Adaptation of the US Army's after-action review for simulation debriefing in healthcare. *Simulation in Healthcare* 8, 6 (2013), 388–397.
[79] Johanna Schmidhuber, Stephan Schlögl, and Christian Ploder. 2021. Cognitive

Load and Productivity Implications in Human-Chatbot Interaction. In *2021 IEEE 2nd International Conference on Human-Machine Systems (ICHMS)*. IEEE, 1–6.

[80] Samuel Sanford Shapiro and Martin B Wilk. 1965. An analysis of variance test for normality (complete samples). *Biometrika* 52, 3/4 (1965), 591–611.

[81] Jesper Simonsen and Morten Hertzum. 2010. Iterative participatory design. *Design research: Synergies from interdisciplinary perspectives* 1 (2010), 16–32.

[82] Igor Steinmacher, Tayana Uchoa Conte, Christoph Treude, and Marco Aurélio Gerosa. 2016. Overcoming open source project entry barriers with a portal for newcomers. In *Proceedings of the 38th International Conference on Software Engineering*. 273–284.

[83] Jiao Sun, Q Vera Liao, Michael Muller, Mayank Agarwal, Stephanie Houde, Kartik Talamadupula, and Justin D Weisz. 2022. Investigating explainability of generative AI for code through scenario-based design. In *27th International Conference on Intelligent User Interfaces*. 212–228.

[84] Silvia Tamayo-Moreno and Diana Pérez-Marín. 2017. Designing and evaluating pedagogic conversational agents to teach children. *International Journal of Educational and Pedagogical Sciences* 11, 3 (2017), 521–526.

[85] Priyan Vaithilingam, Tianyi Zhang, and Elena L Glassman. 2022. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In *CHI Conference on Human Factors in Computing Systems extended abstracts*. 1–7.

[86] Stefano Valtolina, Barbara Rita Barricelli, and Serena Di Gaetano. 2020. Communicability of traditional interfaces VS chatbots in healthcare and smart home domains. *Behaviour & Information Technology* 39, 1 (2020), 108–132.

[87] Matthew Verleger and James Pembridge. 2018. A pilot study integrating an AI-driven chatbot in an introductory programming course. In *2018 IEEE Frontiers in Education conference (FIE)*. IEEE, 1–4.

[88] Ari Ezra Waldman. 2020. Cognitive biases, dark patterns, and the 'privacy paradox'. *Current opinion in psychology* 31 (2020), 105–109.

[89] MS Walgama and B Hettige. 2017. Chatbots: The next generation in computer interfacing–A Review. (2017).

[90] Ruotong Wang, Ruijia Cheng, Denae Ford, and Thomas Zimmermann. 2023. Investigating and Designing for Trust in AI-powered Code Generation Tools. *arXiv preprint arXiv:2305.11248* (2023).

[91] Alexandra Weidemann and Nele Rußwinkel. 2021. The Role of Frustration in Human–Robot Interaction–What Is Needed for a Successful Collaboration? *Frontiers in Psychology* (2021), 707.

[92] Matt Welsh. 2022. The End of Programming. *Commun. ACM* 66, 1 (2022), 34–35.

[93] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C Schmidt. 2023. A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382* (2023).

[94] Sebastian Wollny, Jan Schneider, Daniele Di Mitri, Joshua Weidlich, Marc Rittberger, and Hendrik Drachsler. 2021. Are we there yet?-A systematic literature review on chatbots in education. *Frontiers in Artificial Intelligence* 4 (2021), 654924.

[95] Austin P Wright, Zijie J Wang, Haekyu Park, Grace Guo, Fabian Sperrle, Mennatallah El-Assady, Alex Endert, Daniel Keim, and Duen Horng Chau. 2020. A comparative analysis of industry human-AI interaction guidelines. *arXiv preprint arXiv:2010.11761* (2020).

[96] Frank F Xu, Uri Alon, Graham Neubig, and Vincent Josua Hellendoorn. 2022. A systematic evaluation of large language models of code. In *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming*. 1–10.

[97] Frank F Xu, Bogdan Vasilescu, and Graham Neubig. 2022. In-IDE code generation from natural language: Promise and challenges. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 31, 2 (2022), 1–47.

[98] Daniel M Yellin. 2023. The Premature Obituary of Programming. *Commun. ACM* 66, 2 (2023), 41–44.