Toward Inclusive AI-Driven Development: Exploring Gender Differences in Code Generation Tool Interactions

Manaal Basha[1], Ivan Beschastnikh[1], Gema Rodriguez-Perez[1], Cleidson R. B. de Souza[2]
[1]University of British Columbia [2]Federal University of Pará
manaals@student.ubc.ca, bestchai@cs.ubc.ca,
gerope@mail.ubc.ca, cleidson.desouza@acm.org

***Abstract*—Context: The increasing reliance on Code Generation Tools (CGTs), such as Windsurf and GitHub Copilot, are revamping programming workflows and raising critical questions about fairness and inclusivity. While CGTs offer potential productivity enhancements, their effectiveness across diverse user groups have not been sufficiently investigated.**

Objectives: We hypothesize that developers' interactions with CGTs vary based on gender, influencing task outcomes and cognitive load, as prior research suggests that gender differences can affect technology use and cognitive processing.

Methods: The study will employ a mixed-subjects design with 54 participants, evenly divided by gender for a counterbalanced design. Participants will complete two programming tasks (medium to hard difficulty) with only CGT assistance and then with only internet access. Task orders and conditions will be counterbalanced to mitigate order effects. Data collection will include cognitive load surveys, screen recordings, and task performance metrics such as completion time, code correctness, and CGT interaction behaviors. Statistical analyses will be conducted to identify statistically significant differences in CGT usage.

Expected Contributions: Our work can uncover gender differences in CGT interaction and performance among developers. Our findings can inform future CGT designs and help address usability and potential disparities in interaction patterns across diverse user groups.

Conclusion: While results are not yet available, our proposal lays the groundwork for advancing fairness, accountability, transparency, and ethics (FATE) in CGT design. The outcomes are anticipated to contribute to inclusive AI practices and equitable tool development for all users.

***Index Terms*—Code Generation Tools, Cognitive Load, Gender Differences, Programming Task Performance, Human-Computer Interaction.**

I. Introduction

Code Generation Tools (CGTs) based on Largue-Language Models (LLMs), such as Windsurf[1] and GitHub Copilot[2], are at the forefront of software development with millions of paid users [1]. Studies on user perceptions of CGTs highlight several of their benefits, especially for novice users [2]. However, challenges remain in building trust in these tools [3], [4] and addressing issues related to debugging [5], which users often struggle with. Over-reliance on CGTs and distracting tool features [6] have also been identified as significant concerns. Additionally, there is a general fear that CGTs may introduce vulnerabilities into the code [7], [8]. Despite these concerns, CGTs are designed to reduce cognitive load during certain tasks [9], [10].

Significant progress has been made in addressing these issues, with efforts underway to develop more transparent tool features [11], [12], improve tool accuracy through recent model enhancements [13], and promote responsible AI practices by establishing and adhering to standardized guidelines [14], [15]. Our work aligns with the fairness and transparency dimensions of the FATE[3] (Fairness, Accountability, Transparency, Ethics) framework by investigating potential biases in CGT usability, focusing on gender disparities in task performance and tool interactions, with the aim of informing more equitable tool design. There has been prior work on users with different language backgrounds [16], [17], [18]. But, there is much to be explored with regards to *gender* differences in CGT usage.

Addressing gender differences in CGT tools is particularly important as gender diversity is a key factor in ensuring that these tools serve a broad range of users effectively. The training data used by CGTs often reflect existing biases; for example, GPT-3 has been shown to exhibit biases due to the text it was trained on [19]. Similarly, the code and comments used to train CGTs are predominantly sourced from a male-dominated industry [20], which may lead these tools to unintentionally cater to a narrow demographic. This could reinforce existing biases and limit the tools' accessibility and usefulness for other groups. We hypothesize that gender differences exist in how CGTs, trained on datasets predominantly created by men developers, are used. This study investigates the influence of gender on CGT usage not to generalize or reinforce divisions, but to reveal mismatches between tool design and user needs. Insights from our work will inform the development of CGTs that are more inclusive and responsive to the cognitive and behavioral diversity across developer populations. Particularly

[1]https://windsurf.com/
[2]https://github.com/features/copilot
[3]https://www.microsoft.com/en-us/research/theme/fate/

in settings where team composition cannot change, but the tools can be made more accommodating.

To validate our hypothesis, we propose to evaluate gender difference in CGTs in terms of cognitive load, task outcome, and CGT utilization as compared to the standard alternative of using the internet. This comparison is necessary to establish a benchmark for how gender differences manifest in a familiar and widely-used alternative to CGTs. Forums, documentation, and search engines are resources from internet that developers often rely on for programming support. By comparing gender differences in cognitive load, task outcomes, and utilization between CGTs and internet resources, we can better understand whether CGTs offer unique challenges or advantages that differ from more traditional methods, and identify if they present gender-related disparities.

II. Motivation

CGTs are now integrated into most modern Integrated Development Environments (IDEs), offering features like built-in auto-completion, as seen in IntelliSense for VS Code and IntelliJ. Advanced paid versions, such as GitHub Copilot and Tabnine, have further normalized CGT usage in software development, aiming to enhance productivity. However, this widespread adoption also raises questions concerning FATE.

Terrell et al. [21] highlighted the presence of implicit bias in technical domains by showing that women's contributions to open-source projects are accepted at higher rates when their gender is not identifiable, but at significantly lower rates when it is known. These findings suggest the existence of systemic barriers that are not immediately visible or addressed. The authors concluded that more efforts are needed to create a more inclusive environment for diverse user groups. More recently, Treude and Hata [22], using a back-translation approach, identified a significant disparity in the gendered pronoun associations with different software development tasks. For instance, requirements activities were associated 6% of the time with men, while testing activities were 100% associated with men.

To provide a structured approach to evaluate gender inclusivity, Burnett et al. introduced the GenderMag framework [23]. They identified five cognitive facets known to differ by gender: attitude toward risk, self-efficacy, information processing style, motivations, and learning preferences. For example, women tend to exhibit lower self-efficacy and higher risk aversion when engaging with technology [23], [24], which may influence how they interact with CGTs.

These factors have been shown to significantly impact decision-making and interactions with technology. For instance, risk-averse individuals may hesitate to explore or adopt new tools like CGTs, while those who are more risk-tolerant are likely to experiment and engage more freely. This pattern was observed by Durndell and Haag [25] in a study involving 150 participants during the early days of internet adoption. The study assessed participants' experiences using various tests and revealed notable gender differences. Half of the participants were female, and compared to males, they reported spending less time using the internet, holding less positive attitudes toward it, experiencing greater computer-related anxiety, and demonstrating lower computer self-efficacy.

Additionally, the automation inherent in CGTs can influence users' learning styles. For instance, CGTs often suggest code snippets, making them particularly useful to users who prefer exploratory or tinkering-based learning approaches. These users may experiment with the suggestions, using trial and error to refine the code and resolve bugs, ultimately aiming to produce correct and functional code. However, research has shown that women tend to exhibit less interest and confidence in tinkering behaviors [26], which are critical for maximizing the potential of CGTs.

This distinction aligns with findings from Barke et al. [27] who identified two distinct user states in their study of CGT interaction trends. Users in an *exploratory* state engaged in tinkering, experimenting with tool features, and using trial-and-error methods to navigate coding challenges. In contrast, users in an *acceleration* state leveraged CGTs to enhance their performance, applying the tools efficiently because they already knew what to code and how to implement it. These differing approaches highlight how user traits and learning preferences can shape effective use of CGTs.

Gender differences in cognitive load have also been observed in various learning contexts, shedding light on how technology design and interaction types can affect user experiences. For instance, Chen et al. [28] studied the cognitive load of school-aged children using learning robots and found that boys experienced significantly lower cognitive load than girls, despite no significant difference in their learning performance. The boys appeared to benefit more from the robots, which the authors attributed to the robot interface design favoring visual and spatial elements. These elements have been shown to align better with male cognitive strengths, while females tend to benefit more from auditory components [29]. Such insights highlight the importance of examining how CGT interfaces might inadvertently cater to certain cognitive strengths, potentially influencing their accessibility and effectiveness for different genders.

Building on prior studies that highlight gender-based differences in technology use and adoption [23], [25], we hypothesize that similar disparities exist in how users interact with CGTs. Research has shown gendered differences in cognitive load across learning environments, particularly when interfaces align with certain sensory modalities [28], [29], motivating our investigation into whether CGTs similarly create unequal cognitive demands (RQ1). Further, women have been found to engage less in exploratory or tinkering-based programming behaviors [26], which are linked to CGT effectiveness [27], raising questions about gender-based differences in both task outcomes (RQ2) and tool utilization (RQ3). This study ultimately aims to provide evidence that can guide the design of more inclusive and equitable CGTs, ensuring they better serve a diverse user base.

III. Research Questions and Hypotheses

This study aims to evaluate gender differences in the use of CGTs during programming tasks as compared to the standard approach of using the internet, focusing on cognitive load, task performance, and interaction patterns. Based on the prior studies described in the previous section, we defined the following research questions to guide this investigation:

RQ1: Are there significant gender differences in the cognitive load (intrinsic, extraneous, and germane load scores) experienced by participants when using CGTs to complete programming tasks? Cognitive load can affect how developers interact with CGTs and their overall performance. This question investigates whether gender influences cognitive load during the use of CGTs.

H1: There will be a statistically significant difference in each type of cognitive load between participants of different genders when using CGTs [23], [28].

RQ2: How does CGT outcomes differ between participants of different genders? This question evaluates whether gender impacts task outcomes (i.e., correctness and completion time) when using CGTs.

H2: There will be statistically significant differences in task outcomes between participants of different genders when using CGTs [27].

RQ3: How does CGT utilization differ between participants of different genders? This question evaluates whether gender impacts tool utilization (i.e., proportion of final code that is CGT-generated, suggestion acceptance rate (SAR), generated code that was modified after being accepted, prompt generation count) when using CGTs.

H3: There will be statistically significant differences in tool utilization between participants of different genders when using CGTs [24], [25], [26], [27].

H4: There will be a statistically significant difference in tool utilization when using a CGT compared to the internet approach [29].

IV. Methodology

This section outlines the experimental design, participant recruitment, task selection, counterbalancing strategies [30], and metrics used in the study to explore gender differences in utilizing CGTs. The goal is to investigate how the cognitive load and task performance differ between genders when using CGTs like Windsurf and compare it with traditional programming practices using the internet.

A. Participants:

To ensure meaningful results from our sample size, we performed power analysis using [31] as our guide, and a medium effect size of approximately 0.4 or higher, meaning we expect a moderate to large difference in our study which is detectable and noticeable. This aligns with our study hypothesis. Using an alpha of 0.05 and 95% power, meaning a 95% chance of correctly detecting an effect if it exists, we would need approximately 54 participants for a best case scenario in a between subjects design. Therefore, we aim to recruit 27 participants per gender using posters and social media. For inclusion criteria, participants should be graduate students or students in their final year of undergraduate studies in computer science or related fields. They should have prior programming experience, with a focus on Python, and be adept with basic concepts of coding. Participants with no prior experience using CGTs or Python will be excluded to maintain consistency in the sample's ability to complete the tasks. Participants who do not self-identify as women or men will be included only if the sample size is sufficient to ensure meaningful analysis. Participants will complete a pre-study survey to ensure study requirements are met, and a demographic survey to determine key information like CGT experience level, coding level, self-identify gender, and age which may relate to participant results. A post-study user experience survey will also be given to participants to extract subjective information that can better help us understand our results.

B. Experimental Design:

The study will employ a mixed-subjects design, incorporating both within-subjects (task type and CGT condition) and between-subjects (gender) variables. Each participant will complete two Python programming tasks: one using a CGT and one using the internet. Windsurf, a free and accessible CGT that integrates with VS Code and supports both auto-suggestions and chat-based prompting, will be used due to its availability, cost-effectiveness, and ease of integration into developer workflows.

Participants will use their own laptops in a private lab space, ensuring familiarity with their development environment and reducing variability from unfamiliar hardware. This within-subjects structure reduces individual difference effects, as each participant acts as their own control.

To evaluate the effects of gender and CGT usage on cognitive load, task performance, and tool utilization, we adopt a semiparametric modeling approach that accommodates the varied data types in our study. Specifically, we leverage ordinal regression models, such as the Proportional Odds Model (POM), as well as generalized linear models (GLMs) suited to continuous outcomes, including beta regression for proportional values. Our outcome variables Y include: ordinal (ICL, ECL, GCL on a 9-point scale), count (prompt generation count), continuous (task time in seconds), and proportion (suggestion acceptance rate, CGT-generated code ratio, code modification rate, and correctness ratios) data types.

Ordinal response variables will be analyzed using POM, because it does not assume normality and is robust to extreme values. POM is particularly appropriate for Likert-type outcomes and allows us to estimate how covariates (e.g., gender, task, tool condition) influence cognitive load.

Continuous outcomes (e.g., task completion time) will be analyzed using robust linear regression to accommodate skewness and reduce sensitivity to non-normal errors. For count data (e.g., prompt count), we will use Poisson or negative

binomial regression, based on dispersion tests. Proportion outcomes (e.g., suggestion acceptance rate, code correctness) will be scaled to (0,1) interval and modeled using beta regression.

To account for repeated measures (e.g., multiple tasks per participant), we will include random intercepts for participant ID where appropriate, forming mixed-effects extensions of the models listed above. This modeling framework offers a unified approach to handling multiple outcome types, improving the interpretability and comparability of results while avoiding strong distributional assumptions. This strategy enables a consistent and statistically robust framework for hypothesis testing across all dependent variables.

To reduce order effects and increase internal validity, the order in which tasks and conditions are presented will be counterbalanced. Participants will be randomly assigned to one of two counterbalanced groups as seen in Table I.

TABLE I
COUNTERBALANCED TASK AND CGT CONDITIONS

CGT / Tasks	**Starts With CGT**	**Starts Without CGT**
Task 1 → Task 2	Group A1 (With → Without)	Group B1 (Without → With)
Task 2 → Task 1	Group A2 (With → Without)	Group B2 (Without → With)

By randomly assigning participants to one of these groups, we aim to control for the effects of task order, ensuring that both conditions (CGT vs. internet) are presented to participants in a randomized sequence.

For measuring cognitive load, a 15 question survey utilizing a 9-point Likert-type scale ranging from 1 (not at all applicable) to 9 (fully applicable) was selected based on a meta-analysis by [32] which found that the use of this scale in the existing cognitive load questionnaire was linked to higher reliability values. The questionnaire has also been shown to have strong internal consistency (e.g., Cronbach's $\alpha > .80$) and construct validity across multiple software and learning environments. We allocate 5 survey questions for each cognitive load type, and base these on the recommended questionnaire found in [32]. Each participant will complete the full survey after each task to better understand the relationship between different aspects of cognitive load and CGT usage.

1) Task Selection: The two tasks in this study were designed to test participants' Python programming skills, focusing on real-world applicability. These tasks were specifically chosen to be of medium to hard difficulty, aiming to challenge graduate students or final-year undergraduate students in computer science. Each task was designed by the authors to be completed within approximately 60 minutes, and this duration was confirmed through preliminary testing on a group of 5 participants. Participants in this study will be given a maximum of 5 hours to complete the tasks. Both tasks were evaluated based on their cyclomatic complexity and lines of code (LOC) to ensure that they are appropriately challenging but still manageable in the given time frame. Task 1 has a cyclomatic complexity of 15 and approximately 70 LOC, while Task 2 has a cyclomatic complexity of 14 and approximately 80 LOC. The tasks were designed with real-world scenarios in mind, which participants are likely to encounter in their future careers.

Both tasks were intentionally designed to be original and not easily solvable with a quick Google search or by prompting the CGT. This ensures that participants engage in problem-solving rather than relying on prior knowledge or readily available answers. The tasks also include constraints and requirements that encourage participants to write well-structured, efficient code and test it against various edge cases. To guide participants, a template will be provided for each task, which includes function names with expected function inputs and return values as a docstring, as well as key libraries to use. This was influenced by LeetCode[4] coding problems which provides a similar layout. This ensures consistency across participants' approaches in terms of layout while still allowing for individual problem-solving and coding creativity. Further, this allows for later metrics such as unit tests for code correctness and functionality to be more reliable and easier across all submissions. Tasks will be published later to avoid potential participants seeing them or being able to find the problems.

C. RQ1: Cognitive Load Measurement:

To evaluate cognitive load, participants will complete a survey after each task. These questions will measure three different types of cognitive load as defined by Krieglstein et al. [33]:

- **Intrinsic Load (ICL):** The cognitive effort required due to the complexity of the task itself.
- **Extraneous Load (ECL):** The cognitive effort imposed by the environment or interface (e.g., distractions, poorly designed tool interfaces).
- **Germane Load (GCL):** The cognitive effort involved in creating and maintaining schemas or mental models while solving the task.

These cognitive load scores will be measured using an optimized questionnaire from [33] after each task and used to assess differences in cognitive load across gender, particularly in relation to the use of CGTs and the use of the internet. Alternative measures like the NASA Task Load Index (TLX) questionnaire was considered but based on a construct validity study [34] with ten adults, researchers found that the questionnaire lacked sensitivity to personal capacities or task demands suggesting that the measure has limited utility. The authors advise that the approach is best when task demands are clearly perceptible. However, NASA advised that redefining the existing generic TLX metric labels to a relative situation can be a great strategy, however, these modified approaches need to be validated to ensure their reliability and sensitivity [35]. Krieglstein et al. [33] reviewed and validated the cognitive load questionnaire used in this study by using 54 participants.

[4]https://leetcode.com/

The authors confirmed that the different types of cognitive load can be measured in a reliable and valid manner with the proposed questionnaire. The reason we differentiate the sources of cognitive load in this study is to better isolate the sources of the effects.

D. RQ2: Task Outcome Metrics

Participants will have their screen recorded to be used in combination with the Code Watcher extension. We will analyze the following metrics:

- **Task Completion Time:** Time taken to complete each task. This will be measured from the moment the task starts until the participant submits their solution. This will be calculated using the participant recordings.
- **Core Code Correctness:** Correctness focuses solely on whether the code meets the essential requirements of the task for predefined, straightforward inputs. The inputs and expected outputs will be the same as those provided in the user task instructions as examples. It will be assessed as a percentage of test cases successfully passed. If the code produces the expected output for all specified unit test inputs, it will be deemed correct. This ensures a baseline evaluation of the core logic, regardless of how well the code handles variations or additional nuances.
- **Advanced Code Correctness:** Functionality refers to how comprehensively the code meets the full range of task requirements, including handling edge cases and nuances beyond the core problem. Functionality will be objectively measured by running the submitted code against an extensive set of test cases. These test cases will include both typical scenarios and challenging edge cases, such as unusual input ranges, empty or null values, and high-volume data inputs. Functionality will be quantified as the percentage of test cases successfully passed, indicating how well the code addresses diverse and practical aspects of the problem.

E. RQ3: CGT Usage Metrics

To assess how CGT utilization impacts task performance, we will use an in-lab-built tool called Code Watcher. The tool is a custom VS Code extension that is specifically designed to monitor and record participants' coding interactions in real time. By seamlessly integrating into the development environment, it provides detailed data on how users engage with CGTs during tasks. More technical information regarding Code Watcher can be found in our repository.

In this study, we will collect and analyze the following metrics:

- **Prompt Generation Count:** The number of times the CGT is prompted to generate code in the IDE specifically in the chat window. This will measure how actively participants rely on the CGT to produce solutions. This is different than the auto-suggestion feature that recommends code as the user types. This will be measured by analyzing user recordings.
- **Suggestion Acceptance Rate (SAR):** The percentage of generated code suggestions accepted by the participants, indicating how often they find the CGT's suggestions useful or usable. This will be measured using Code Watcher data.
- **Modification Rate:** How often participants modify the code suggested by the CGT and that was accepted by the participants (e.g., changing variable names). This will be measured using Code Watcher data.
- **Proportion of Final Code Generated by CGT:** The percentage of code in the participant's final submission that is directly generated by the CGT compared to code written by the participants themselves. More specifically, the ratio of LOC that are AI written and the total LOC. This will be measured using Code Watcher data.

These metrics will be compared across gender to determine if there are any significant differences in how participants of different genders interact with CGTs. Code Watcher's performance will also be validated through manual confirmation using participant screen recordings.

F. Ethical Considerations

In order to ensure a safe and comfortable environment for all participants, we will ensure informed consent, confidentiality, and provide debriefs before and after the study. Ethics approval will be sought from our university, and the study will not start until approval is granted.

- **Informed Consent:** Participants will be fully informed about the study's objectives and procedures, and written consent will be obtained before data collection begins. Participants can also end their involvement in the study at any time without penalty.
- **Confidentiality:** Participants' personal information and data will be kept confidential, and all recordings will be anonymized.
- **Debriefing:** At the end of the study, participants will be debriefed about the research objectives and results, and any questions will be answered.
- **Compensation:** Participants will receive a $25 gift card after their completion of the study.

G. Threats to Validity

Internal Validity: In order to ensure that our observed effects are due to changes in our independent variables and not from external factors we will use a within-subjects design which controls for individual difference and improves comparability. Further, prior programming experience is accounted for to ensure that participants do not lack a computing background which can effect their overall task performance. All participants complete the same task under the same conditions to reduce variability in problem difficulty or environmental factors. To mitigate order effects, we counterbalance task order across participants and include breaks in the session to reduce fatigue. In addition, we acknowledge the potential for subjective bias for survey submissions and complement this with performance-based metrics. Gender perception of

task difficulty can lead to differences in perceived difficulty and could influence task performance. We account for this by measuring task ICL, which helps determine if there is any effect of gender on task difficulty. Participants will be informed that screen recordings are used strictly for analyzing CGT interaction patterns, not performance evaluation, to reduce observation bias.

External Validity: To improve our findings generalization to real-world scenarios, we ensure that tasks are designed to reflect real- world software development challenges, making findings more applicable to professional developers. Further, we are selecting participants in their final year of undergraduate study and above as they are more likely to have exposure to internships and co-ops, ensuring industry-relevant skills. However, as our sample consists of students, our findings may not fully capture industry professional behaviors and should be addressed in the future. Additionally, this study focuses on a specific CGT, results may not generalize to all CGT tools. Although this study uses Windsurf as the representative CGT, its prompt-based, inline suggestion paradigm is shared across tools like GitHub Copilot. As such, we expect our findings to generalize to similar tools, though future work should validate these results across different CGT interfaces and programming environments. Further, regional and cultural factors may influence results, and future studies should consider an even larger more diverse participant pool.

Construct Validity: The metrics used in this study such as task completion time and task ICL aim to allow us to better understand the cognitive effort and difficulty, but they may not fully capture subjective workload or user experience. Additionally, differences in familiarity with VS Code or Windsurf may introduce unintended variability in performance.

V. Preliminary Analysis

We conducted a pilot study with five participants (3 men and 2 women) to validate and refine the proposed study methodology. Group A consisted of 1 woman and 2 men, Group B consisted of 1 woman and 1 man. The pilot study aimed to assess the clarity of the tasks, evaluate the utility of the data collection tool, and determine whether the proposed metrics and tests could be successfully implemented. Feedback from the participants indicated that the tasks were generally clear and appropriately challenging, though minor revisions were suggested to improve the clarity of the constraints and expected outcomes. Based on this feedback, we plan to provide additional examples and detailed explanations in the task templates to ensure all participants understand the requirements fully. Code Watcher successfully captured user interactions within the VS Code IDE and flagged code that was potentially generated by the CGT. The data collected included timestamps, user edits, and flagged AI-generated code, providing a comprehensive view of participants' interactions with the CGT. The proposed metrics, such as task completion time, proportion of AI-generated code, and code correctness, were successfully calculated. Unit tests ran successfully, confirming that the correctness and functionality metrics could be reliably evaluated. However, one minor deviation from the original plan involved the need for additional preprocessing steps to handle inconsistencies in the way participants interacted with the task templates, such as variations in naming conventions for functions. These preprocessing steps will be standardized in the full study to streamline data analysis. Preliminary descriptive statistics from the pilot provide an initial sense of the task performance metrics and cognitive load ratings. Mean task completion times across all participants were 80 minutes for Task 1 and 75 minutes for Task 2. Cognitive load ratings on the 9-point Likert scale varied across load types and conditions. For the CGT condition, participants reported an average ICL of 2.1 and 4.1 for Tasks 1 and 2, respectively. These metrics will be monitored closely in the main study to assess variability across participants. In summary, the pilot study confirmed that the experimental design, data collection tool, and proposed metrics are effective and feasible. Adjustments will be made to the task templates and preprocessing steps based on pilot feedback to improve clarity and streamline data analysis in the full study.

All the tasks, surveys, and explanations about the experiment can be found here.

VI. Contribution and Implications

This study's primary contribution is to examine gender differences in the interaction with and effectiveness of CGTs. By analyzing cognitive load and task outcomes across gender groups and with/without CGT usage, we aim to uncover user differences that can guide the future design and implementation of CGTs.

We also seek to contribute to the broader field of FATE in AI4SE by addressing the underexplored area of gender differences in CGT usage. Our analysis of cognitive load and performance metrics will provide insights into how CGTs may differently affect demographic groups, helping inform the development of more inclusive designs that meet the needs of a wider user base.

In general, our research could influence future user-centric design principles by encouraging CGT developers to integrate gender-aware features that reduce cognitive load or improve usability for underrepresented groups. This is aligned with suggestions proposed by Wang et al. [36], [4], who proposed and evaluated different designs to increase human trust in CGT. However, as they recognized, they had a biased sample. This is also aligned with Burnett's GenderMag method [23] that aims to find and fix gender-inclusivity "bugs".

Ultimately, our findings could assist businesses in assessing challenges related to user acceptance and guide strategies for more effective tool development and user education. This would contribute to a more inclusive development ecosystem where CGTs are optimized to serve users from diverse backgrounds.

References

[1] D. Ramel, "Copilot by the numbers: Microsoft's big ai bet paying off," May 2024. [Online]. Available: https://visualstudiomagazine.com/Articles/2024/02/05/copilot-numbers.aspx

[2] J. Prather, B. N. Reeves, J. Leinonen, S. MacNeil, A. S. Randrianasolo, B. A. Becker, B. Kimmel, J. Wright, and B. Briggs, "The widening gap: The benefits and harms of generative ai for novice programmers," in *Proceedings of the 2024 ACM Conference on International Computing Education Research-Volume 1*, 2024, pp. 469–486.

[3] R. Cheng, R. Wang, T. Zimmermann, and D. Ford, ""it would work for me too": How online communities shape software developers' trust in ai-powered code generation tools," *ACM Transactions on Interactive Intelligent Systems*, vol. 14, no. 2, pp. 1–39, 2024.

[4] R. Wang, R. Cheng, D. Ford, and T. Zimmermann, "Investigating and designing for trust in ai-powered code generation tools," in *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, ser. FAccT '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 1475–1493.

[5] D. Jayagopal, J. Lubin, and S. E. Chasins, "Exploring the learnability of program synthesizers by novice programmers," in *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*, 2022.

[6] J. Prather, B. N. Reeves, P. Denny, B. A. Becker, J. Leinonen, A. Luxton-Reilly, G. Powell, J. Finnie-Ansley, and E. A. Santos, ""it's weird that it knows what i want": Usability and interactions with copilot for novice programmers," *ACM Transactions on Computer-Human Interaction*, vol. 31, no. 1, p. 1–31, Nov 2023.

[7] H. Pearce, B. Ahmad, B. Tan, B. Dolan-Gavitt, and R. Karri, "Asleep at the keyboard? assessing the security of github copilot's code contributions," in *2022 IEEE Symposium on Security and Privacy (SP)*, May 2022.

[8] A. Rajbhoj, A. Somase, P. Kulkarni, and V. Kulkarni, "Accelerating software development using generative ai: Chatgpt case study," in *Proceedings of the 17th Innovations in Software Engineering Conference*, ser. ISEC '24. New York, NY, USA: Association for Computing Machinery, 2024.

[9] T. Zhang, L. Lowmanstone, X. Wang, and E. L. Glassman, "Interactive program synthesis by augmented examples," in *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, 2020.

[10] W. Mendes, S. Souza, and C. R. B. De Souza, ""you're on a bicycle with a little motor": Benefits and challenges of using ai code assistants," in *2024 IEEE/ACM 17th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE)*, 2024, pp. 144–152.

[11] I. Tenney, R. Mullins, B. Du, S. Pandya, M. Kahng, and L. Dixon, "Interactive prompt debugging with sequence salience," *arXiv preprint arXiv:2404.07498*, 2024.

[12] R. Patel, "Introducing code referencing for github copilot chat in visual studio," Dec 2024. [Online]. Available: https://devblogs.microsoft.com/visualstudio/introducing-code-referencing-for-github-copilot-chat-in-visual-studio/

[13] J. Liu, C. S. Xia, Y. Wang, and L. Zhang, "Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[14] F. Leboukh, E. B. Aduku, and O. Ali, "Balancing chatgpt and data protection in germany: challenges and opportunities for policy makers," *Journal of Politics and Ethics in New Technologies and AI*, vol. 2, no. 1, pp. e35 166–e35 166, 2023.

[15] M. Constantinides, E. Bogucka, D. Quercia, S. Kallio, and M. Tahaei, "Rai guidelines: Method for generating responsible ai guidelines grounded in regulations and usable by (non-) technical roles," *Proceedings of the ACM on Human-Computer Interaction*, vol. 8, no. CSCW2, pp. 1–28, 2024.

[16] A. Buscemi, "A comparative study of code generation using chatgpt 3.5 across 10 programming languages," *arXiv preprint arXiv:2308.04477*, 2023.

[17] Z. Wang, G. Cuenca, S. Zhou, F. F. Xu, and G. Neubig, "Mconala: a benchmark for code generation from multiple natural languages," *arXiv preprint arXiv:2203.08388*, 2022.

[18] K. Koyanagi, D. Wang, K. Noguchi, M. Kondo, A. Serebrenik, Y. Kamei, and N. Ubayashi, "Exploring the effect of multiple natural languages on code suggestion using github copilot," in *Proceedings of the 21st International Conference on Mining Software Repositories*, ser. MSR '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 481–486. [Online]. Available: https://doi.org/10.1145/3643991.3644917

[19] S. Gururangan, D. Card, S. K. Dreier, E. K. Gade, L. Z. Wang, Z. Wang, L. Zettlemoyer, and N. A. Smith, "Whose language counts as high quality? measuring language ideologies in text data selection," *arXiv preprint arXiv:2201.10474*, 2022.

[20] S. Daniel, R. Agarwal, and K. J. Stewart, "The effects of diversity in global, distributed collectives: A study of open source project success," *Information Systems Research*, vol. 24, no. 2, pp. 312–333, 2013.

[21] J. Terrell, A. Kofink, J. Middleton, C. Rainear, E. Murphy-Hill, C. Parnin, and J. Stallings, "Gender differences and bias in open source: Pull request acceptance of women versus men," *PeerJ Computer Science*, vol. 3, p. e111, 2017.

[22] C. Treude and H. Hata, " She Elicits Requirements and He Tests: Software Engineering Gender Bias in Large Language Models ," in *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2023, pp. 624–629. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/MSR59073.2023.00088

[23] M. Burnett, S. Stumpf, J. Macbeth, S. Makri, L. Beckwith, I. Kwan, A. Peters, and W. Jernigan, "Gendermag: A method for evaluating software's gender inclusiveness," *Interacting with computers*, vol. 28, no. 6, pp. 760–787, 2016.

[24] K. Beckwith, "A common language of gender?" *Politics & Gender*, vol. 1, no. 1, pp. 128–137, 2005.

[25] A. Durndell and Z. Haag, "Computer self efficacy, computer anxiety, attitudes towards the internet and reported experience with the internet, by gender, in an east european sample," *Computers in human behavior*, vol. 18, no. 5, pp. 521–535, 2002.

[26] M. Burnett, S. D. Fleming, S. Iqbal, G. Venolia, V. Rajaram, U. Farooq, V. Grigoreanu, and M. Czerwinski, "Gender differences and programming environments: across programming populations," in *Proceedings of the 2010 ACM-IEEE international symposium on empirical software engineering and measurement*, 2010, pp. 1–10.

[27] S. Barke, M. B. James, and N. Polikarpova, "Grounded copilot: How programmers interact with code-generating models," *Proceedings of the ACM on Programming Languages*, vol. 7, no. OOPSLA1, pp. 85–111, 2023.

[28] B. Chen, G.-H. Hwang, and S.-H. Wang, "Gender differences in cognitive load when applying game-based learning with intelligent robots," *Educational Technology & Society*, vol. 24, no. 3, pp. 102–115, 2021.

[29] F. Pauls, F. Petermann, and A. C. Lepach, "Gender differences in episodic memory and visual working memory including the effects of age," *Memory*, vol. 21, no. 7, pp. 857–874, 2013.

[30] D. Gergle and D. S. Tan, "Experimental research in hci," in *Ways of Knowing in HCI*. Springer, 2014, pp. 191–227.

[31] M. Brysbaert, "How many participants do we have to include in properly powered experiments? a tutorial of power analysis with reference tables," *Journal of cognition*, vol. 2, no. 1, p. 16, 2019.

[32] F. Krieglstein, M. Beege, G. D. Rey, P. Ginns, M. Krell, and S. Schneider, "A systematic meta-analysis of the reliability and validity of subjective cognitive load questionnaires in experimental multimedia learning research," *Educational Psychology Review*, vol. 34, no. 4, pp. 2485–2541, 2022.

[33] F. Krieglstein, M. Beege, G. D. Rey, C. Sanchez-Stockhammer, and S. Schneider, "Development and validation of a theory-based questionnaire to measure different types of cognitive load," *Educational Psychology Review*, vol. 35, no. 1, p. 9, 2023.

[34] R. D. McKendrick and E. Cherry, "A deeper look at the nasa tlx and where it falls short," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 62, no. 1. SAGE Publications Sage CA: Los Angeles, CA, 2018, pp. 44–48.

[35] S. G. Hart, "Nasa-task load index (nasa-tlx); 20 years later," in *Proceedings of the human factors and ergonomics society annual meeting*, vol. 50, no. 9. Sage publications Sage CA: Los Angeles, CA, 2006, pp. 904–908.

[36] R. Wang, R. Cheng, D. Ford, and T. Zimmermann, "Investigating and designing for trust in ai-powered code generation tools," in *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, ser. FAccT '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 1475–1493. [Online]. Available: https://doi.org/10.1145/3630106.3658984