Understanding And Guiding Student-AI Interaction In Future Programming Education

Ashley Ge Zhang
University of Michigan
Ann Arbor, Michigan, USA
gezh@umich.edu

Yinuo Yang
University of Michigan
Ann Arbor, Michigan, USA
inon@umich.edu

Maryam Arab
University of Michigan
Ann Arbor, Michigan, USA
maryarab@umich.edu

Yan Chen
Virginia Tech
Blacksburg, Virginia, USA
ych@vt.edu

Steve Oney
University of Michigan
Ann Arbor, Michigan, USA
soney@umich.edu

Abstract

With recent advances in generative AI, delivering personalized learning experience in programming education has become more feasible. However, how students use AI can significantly impact their learning outcomes. To help guide student-AI interactions, it is crucial for instructors to establish effective AI usage policies. These policies may vary based on course requirements and students' backgrounds, shaping different views on appropriate and inappropriate AI use in class. While understanding and guiding student-AI interaction is essential, it remains unclear which needs are general to programming education and which are specific to particular topics. This ambiguity makes it challenging to design a system that is both practical and useful in real-world programming courses. In this paper, we identify gaps in existing literature and propose a study to explore instructors' perspectives on students' AI usage. We also introduce a potential system design that allows instructors to monitor student-AI interactions, detect problematic behaviors, and intervene when these interactions conflict with the pedagogical goals.

CCS Concepts

• **Do Not Use This Code → Generate the Correct Terms for Your Paper**; *Generate the Correct Terms for Your Paper*; Generate the Correct Terms for Your Paper; Generate the Correct Terms for Your Paper.

Keywords

Do, Not, Us, This, Code, Put, the, Correct, Terms, for, Your, Paper

ACM Reference Format:

Ashley Ge Zhang, Yinuo Yang, Maryam Arab, Yan Chen, and Steve Oney. 2018. Understanding And Guiding Student-AI Interaction In Future Programming Education. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 4 pages. https://doi.org/XXXXXXX.XXXXXXX

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

AI has revolutionized programming education for both instructors and students, presenting opportunities to deliver personalized learning experiences at scale. Many existing tools have used AI techniques that can comprehend and generate complex text and code, automating tasks such as diagnosing problems in students' code [1, 13], providing personalized feedback for improvement [2], and predicting students' behavior [4, 9, 18, 24]. Although such tools can facilitate and enhance learning, the inappropriate use of them could instead hurt students' learning outcomes [6].

For students, especially novice programmers, if generative AI is to replace their critical thinking in programming problem solving, instead of supporting it, the metacognitive difficulty students' face could become worse and lead to their over-reliance on AI tools [6, 22]. For instance, prior work has identified behavior of students that struggle with AI, including distraction due to AI interference, difficulty in aligning their conceptual model with AI suggestions, and incorporate code from AI [22]. In addition, AI can go against principles in education, such as directly revealing solutions to students and encouraging copy paste without thinking [16]. Thus, preparing both instructors and students for AI-powered education and preparing AI to understand education become important [21]. This leads to the three research questions of this paper: (1) what do instructors think is the appropriate use of AI in programming education? (2) what do instructors think are inappropriate use of AI in programming education? and (3) how to facilitate monitoring and intervening inappropriate student-AI interactions in programming courses?

While prior work has explored students' usage of AI in programming courses [14, 16] and analyzed individual students' behavior, there lacks a universal understanding of what constitutes appropriate and inappropriate usage of AI. This gap exists for several reasons. First, expectations around AI usage vary widely across different courses, depending on specific requirements, learning objectives, the diverse backgrounds of students, and their learning stages, which influence how they interact with AI tools. Second, there is limited understanding of the key dimensions instructors prioritize when evaluating AI usage. For example, if we aim to visualize students' interactions with AI for instructors, it is unclear

which factors — such as the level of independence, the types of AI assistance used, or the timing of AI engagement — are most important for instructors to focus on. Without clarity on these priorities, it is challenging to design tools that effectively support instructors in understanding and guiding appropriate AI use.

To fill this gap, we propose a need-finding study to understand instructors' perspectives on students' usage of AI, focusing on their views of what constitutes appropriate and inappropriate AI use. This study aims to identify the key dimensions instructors care about when evaluating AI interactions, which will inform the design of future tools. Based on insights from the existing literature, we also propose a potential system design that enables instructors to monitor student-AI interactions and intervene when students use AI in appropriate ways. This system serves as a preliminary exploration of the third research question, with specific features and functionalities to be determined based on the outcomes of the need-finding study.

2 RELATED WORK

2.1 AI in Programming Education

AI has revolutionized programming education for both teaching and learning. Educational tools now leverage AI techniques to comprehend and generate complex data, including text and code, automating tasks such as diagnosing issues in students' code [5], providing personalized feedback [23], generating learning materials [10], and producing code examples [15]. However, integrating AI into programming education introduces challenges for both students and instructors [6]. AI-generated content can be inaccurate, leading to misleading suggestions [6], and may conflict with educational principles by directly revealing solutions or encouraging copy-paste behaviors [16]. It is important to prepare both instructors and students for AI-powered education and designing AI systems that align with educational goals [21]. To address AI's reliability issues, researchers have explored using LLMs as teachable agents, enabling novice students to learn programming by guiding these agents through debugging and code correction tasks [11, 20].

2.2 Understanding Student-AI Interaction

Understanding AI's strengths and limitations helps users make informed decisions and better leverage human-AI collaboration [19]. In programming education, this knowledge aids instructors in guiding students' AI use and spotting behaviors that may hinder learning. Amoozadeh et al. studied student-AI interactions in a CS1 course, highlighting issues like student over-reliance on AI tools, even under supervision [3]. Researchers conducted studies to understand how to foster students' AI usage in programming pedagogical settings. Kazemitabaar et al. deployed CodeAid, an LLM-powered assistant, in a semester-long programming course, proposing design principles like leveraging AI's unique strengths, balancing the directness of responses, and ensuring trust, transparency, and control [16] They also explored cognitive engagement techniques, finding that step-by-step problem-solving guidance with interactive AI prompts was particularly effective [15]. While prior work provides insights into students' AI usage patterns and challenges, there is limited understanding of instructors' perspectives on what constitutes appropriate or inappropriate AI use. Analyzing student-AI interactions at scale remains challenging, especially when dealing with extensive LLM outputs [8]. Beyond education, researchers have explored methods to interpret AI outputs for decision-making and model evaluation [12, 17]. In the educational context, Chen et al. [7] developed StuGPTViz, a visual analytics system that helps instructors explore temporal patterns in student interactions with ChatGPT. In programming education, understanding student-AI interactions requires tools that can coherently combine text and code, allowing instructors to trace problem-solving processes and assess prior AI usage to identify inappropriate behaviors.

3 NEED FINDING STUDY

To better understand instructors' views on appropriate and inappropriate AI usage, we propose a need-finding study using surveys and interviews. The survey will provide a broad overview of instructors' perspectives based on their observation in programming courses. Based on the survey responses, we will conduct follow-up interviews with selected participants to gain deeper insight into their attitude toward AI in programming education. We will use thematic analysis to qualitatively code the data, which will inform the design of a system to support understanding and intervening in student-AI interactions.

3.1 Survey Study

Prior work has explored the opportunities and challenges of AI tools in programming education on a small scale [22] or from literature [6]. However, these studies lack generalizability, often focusing on specific course types, regions, or AI tools. Our survey aims to collect responses from a diverse, global audience of programming instructors at all levels. To ensure broad participation, the survey will be concise, encouraging higher response rates and more generalizable findings. We aim to answer two key research questions: (1) what do instructors consider as appropriate uses of AI in programming education? and (2) what do instructors consider as inappropriate uses of AI in programming education?

3.1.1 Participants. The target population is university-level programming instructors teaching courses ranging from introductory to advanced topics. We will include a wide array of programming courses (e.g., Python programming, web development, machine learning) to capture diverse experiences with AI integration in the curriculum. To recruit participants, we will follow these steps:

- **Identify universities**. We will focus on the top 100 universities worldwide, using established rankings (e.g., Times Higher Education) and enrollment data from Wikipedia [1]. This approach balances the selection of schools and ensures access to active programming courses with exposure to AI tools.
- **Identify relevant courses and instructors**. For each selected university, we will use Google [2] and ChatGPT [3] to compile a list of programming-related courses. We will then extract the names and contact information of instructors, along with course details (e.g., course title, level, subject area) to ensure a balanced sample. We will focus on programming

[1] https://en.wikipedia.org/wiki/List_of_largest_universities_and_university_networks_by_enrollment
[2] https://www.google.com/
[3] https://openai.com/index/chatgpt/

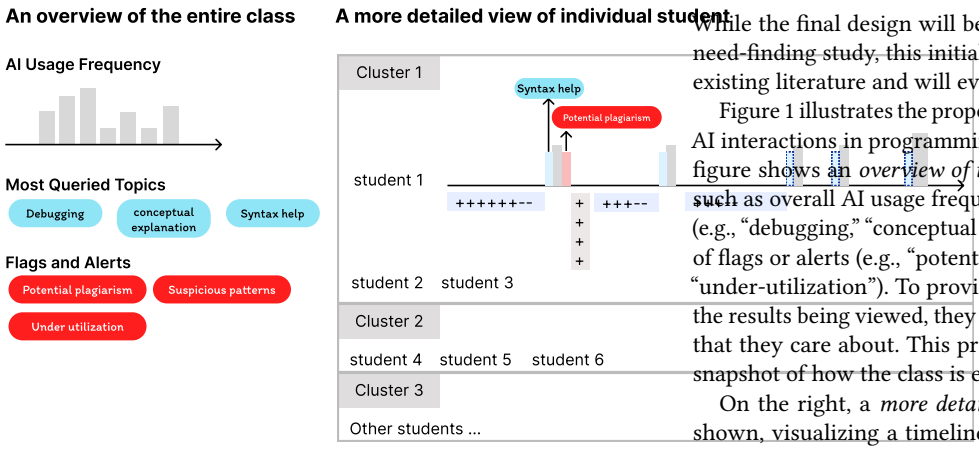

Figure 1: System Design

courses that emphasize hands-on coding and teach programming languages, concept, or practices. Both undergraduate and graduate-level courses will be included. We will exclude seminars, capstone projects, and theory-based courses without substantial coding components. Short-term workshops or bootcamps outside the standard curriculum will also be excluded.

- **Email outreach**. We will send personalized emails to each instructor, including: (1) an description of the study's goals and relevance, (2) a link to the online survey, and (3) a request for referrals, encouraging instructors to forward the invitation or suggest potential participants.

To encourage participation, participants will be entered into a lottery and will receive access to a dataset of anonymized survey responses.

3.1.2 Survey Questionnaire. The survey will gather instructors' perspectives on students' AI usage in programming courses. It includes questions on effective and inappropriate AI use, methods for detecting misuse, institutional AI policies, and the rationale behind these policies. Demographic data such as teaching experience, institution, and course levels will also be collected. To gain deeper insights, participants will have the option to join a 45-minute follow-up interview.

3.2 Interview Study

We will invite interested survey participants for follow-up interviews to explore their perspectives on AI usage in programming courses in more depth. Interviews can help us answer questions that surveys cannot fully address, such as the rationale behind specific policies and nuanced views on student-AI interactions. Additionally, these interviews will inform the iterative design of our system to help instructors understand and intervene in student-AI interactions more effectively.

4 System Design

To help instructors understand and intervene in student-AI interaction, we propose a initial system design shown in Figure 1. While the final design will be refined based on insights from the need-finding study, this initial version addresses gaps identified in existing literature and will evolve through iterative feedback.

Figure 1 illustrates the proposed dashboard for monitoring student–AI interactions in programming education. The left portion of the figure shows an *overview of the entire class*, highlighting metrics such as overall AI usage frequency, most frequently queried topics (e.g., "debugging," "conceptual explanation," "syntax help"), and a set of flags or alerts (e.g., "potential plagiarism," "suspicious patterns," "under-utilization"). To provide instructors with more control on the results being viewed, they can create their own queries or alerts that they care about. This provides instructors with a high-level snapshot of how the class is engaging with AI tools.

On the right, a *more detailed view of an individual student* is shown, visualizing a timeline of their AI queries and code edits. Each bar (or cluster of bars) corresponds to a specific query sent to AI, and color-coding can indicate the type of help requested or the severity of potential issues. Below this timeline, we incorporate a visualization of the student's code edits, showing how much they have inserted or deleted over time. This edit-tracking component helps instructors assess whether a student is making meaningful changes to their code in response to AI outputs or simply copying generated solutions without deeper engagement. The system design supports three key functionalities, described below.

The system design supports the following three key functionalities:

- **Get an overview of students' AI usage.** Instructors can access a class-wide summary of AI interactions through the top-level dashboard (left portion of Figure 1). By consolidating these statistics, the system allows instructors to quickly gauge the overall engagement level and identify which topics may require additional lecture time or clarifications.
- **Be alerted of common patterns and potential problems.** The system automatically flags patterns that could signal academic integrity issues or unproductive usage behaviors.
- **Intervene in when students have inappropriate AI usage.** By drilling down into an individual student's timeline (right portion of Figure 1), instructors can investigate flagged queries and identify whether a student is over-relying on AI-generated code or copying solutions verbatim. Students can be clustered in two ways, either by the queries sent to AI or by their behavior patterns. In this way, the system design not only reveals problematic behaviors but also provides actionable insights to guide instructor-led interventions.

5 Conclusion

This paper identifies gaps in understanding and guiding student-AI interactions in programming education. As generative AI becomes increasingly integrated into educational settings, establishing guidelines to foster meaningful learning while preventing misuses is important. Through our proposed need-finding study, we aim to gather comprehensive insights into instructors' views on appropriate and inappropriate uses of AI, informing the design of tools to support these goals. We propose a system that enables instructors to monitor and intervene in student-AI interactions, ensuring AI

acts as a supportive tool rather than a crutch. By identifying problematic behaviors and enabling timely interventions, the system advances the ongoing conversation about AI integration in education, emphasizing thoughtful approaches that prioritizes student learning and development.

References

[1] 2024. Amazon CodeGuru. https://aws.amazon.com/codeguru.

[2] 2025. Github Copilot. https://github.com/features/copilot.

[3] Matin Amoozadeh, Daye Nam, Daniel Prol, Ali Alfageeh, James Prather, Michael Hilton, Sruti Srinivasa Ragavan, and Amin Alipour. 2024. Student-AI Interaction: A Case Study of CS1 students. In *Proceedings of the 24th Koli Calling International Conference on Computing Education Research*. 1–13.

[4] Ignacio Araya, Victor Beas, Katrina Stamulis, and Héctor Allende-Cid. 2022. Predicting student performance in computing courses based on programming behavior. *Computer applications in engineering education* 30, 4 (2022), 1264–1276.

[5] Rishabh Balse, Viraj Kumar, Prajish Prasad, and Jayakrishnan Madathil Warriem. 2023. Evaluating the quality of llm-generated explanations for logical errors in cs1 student programs. In *Proceedings of the 16th Annual ACM India Compute Conference*. 49–54.

[6] Brett A Becker, Paul Denny, James Finnie-Ansley, Andrew Luxton-Reilly, James Prather, and Eddie Antonio Santos. 2023. Programming is hard-or at least it used to be: Educational opportunities and challenges of ai code generation. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. 500–506.

[7] Zixin Chen, Jiachen Wang, Meng Xia, Kento Shigyo, Dingdong Liu, Rong Zhang, and Huamin Qu. 2024. StuGPTViz: A Visual Analytics Approach to Understand Student-ChatGPT Interactions. *IEEE Transactions on Visualization and Computer Graphics* (2024).

[8] Katy Ilonka Gero, Chelse Swoopes, Ziwei Gu, Jonathan K Kummerfeld, and Elena L Glassman. 2024. Supporting Sensemaking of Large Language Model Outputs at Scale. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 1–21.

[9] Muntasir Hoq, Peter Brusilovsky, Bita Akram, et al. 2024. Explaining Explainability: Early Performance Prediction with Student Programming Pattern Profiling. *Journal of Educational Data Mining* 16, 2 (2024), 115–148.

[10] Xinying Hou, Zihan Wu, Xu Wang, and Barbara J Ericson. 2024. Codetailor: Llm-powered personalized parsons puzzles for engaging support while learning programming. In *Proceedings of the Eleventh ACM Conference on Learning@ Scale*. 51–62.

[11] Hyoungwook Jin, Seonghee Lee, Hyungyu Shin, and Juho Kim. 2024. Teach AI How to Code: Using Large Language Models as Teachable Agents for Programming Education. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 1–28.

[12] Minsuk Kahng, Ian Tenney, Mahima Pushkarna, Michael Xieyang Liu, James Wexler, Emily Reif, Krystal Kallarackal, Minsuk Chang, Michael Terry, and Lucas Dixon. 2024. LLM Comparator: Interactive Analysis of Side-by-Side Evaluation of Large Language Models. *IEEE Transactions on Visualization and Computer Graphics* (2024).

[13] Oscar Karnalim et al. 2021. Promoting code quality via automated feedback on student submissions. In *2021 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–5.

[14] 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.

[15] Majeed Kazemitabaar, Oliver Huang, Sangho Suh, Austin Z Henley, and Tovi Grossman. 2024. Exploring the Design Space of Cognitive Engagement Techniques with AI-Generated Code for Enhanced Learning. *arXiv preprint arXiv:2410.08922* (2024).

[16] Majeed Kazemitabaar, Runlong Ye, Xiaoning Wang, Austin Zachary Henley, Paul Denny, Michelle Craig, and Tovi Grossman. 2024. Codeaid: Evaluating a classroom deployment of an llm-based programming assistant that balances student and educator needs. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 1–20.

[17] Min Hun Lee, Daniel P Siewiorek, Asim Smailagic, Alexandre Bernardino, and Sergi Bermúdez Bermúdez i Badia. 2021. A human-ai collaborative approach for clinical decision making on rehabilitation assessment. In *Proceedings of the 2021 CHI conference on human factors in computing systems*. 1–14.

[18] Enqi Liu, Irena Koprinska, and Kalina Yacef. 2023. Early prediction of student performance in online programming courses. In *International Conference on Artificial Intelligence in Education*. Springer, 365–371.

[19] Duri Long and Brian Magerko. 2020. What is AI literacy? Competencies and design considerations. In *Proceedings of the 2020 CHI conference on human factors in computing systems*. 1–16.

[20] Qianou Ma, Hua Shen, Kenneth Koedinger, and Sherry Tongshuang Wu. 2024. How to teach programming in the ai era? using llms as a teachable agent for debugging. In *International Conference on Artificial Intelligence in Education*. Springer, 265–279.

[21] Francesc Pedro, Miguel Subosa, Axel Rivas, and Paula Valverde. 2019. Artificial intelligence in education: Challenges and opportunities for sustainable development. (2019).

[22] James Prather, Brent N Reeves, Juho Leinonen, Stephen MacNeil, Arisoa S Randrianasolo, Brett A Becker, Bailey Kimmel, Jared Wright, and Ben Briggs. 2024. 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*. 469–486.

[23] Xiaohang Tang, Sam Wong, Marcus Huynh, Zicheng He, Yalong Yang, and Yan Chen. 2024. SPHERE: Scaling Personalized Feedback in Programming Classrooms with Structured Review of LLM Outputs. *arXiv preprint arXiv:2410.16513* (2024).

[24] Stav Tsabari, Avi Segal, and Kobi Gal. 2023. Predicting Bug Fix Time in Students' Programming with Deep Language Models. *International Educational Data Mining Society* (2023).

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009