A Competency Framework for Assessing AI-Assisted Programming Work

Eli Young
eliyoung4now@gmail.com
AlgoArena
Virginia Beach, VA, USA

Andy Luu
andyhluu23@gmail.com
AlgoArena
Virginia Beach, VA, USA

Michael Wehar*
mwehar@haverford.edu
AlgoArena
Virginia Beach, VA, USA

Abstract

AI coding assistants have moved from autocomplete to agents that generate, run, and revise whole programs. As a result, software is increasingly being built by stating intent rather than writing code by hand. Furthermore, students are using these tools in their coursework. In an introductory course, 84 percent of students stated they used ChatGPT for their assignments, exposing a gap in assessment when these tools are at play. When an LLM writes the code, passing test suites alone does not show how well the student works with AI, because there are other expertise-revealing aspects of software development, like security. What separates a strong student (when AI is permitted) is how well they specify a task, direct an agent, and catch bad output. We present a competency framework that scores AI-assisted programming work along five competencies (problem solving, planning, prompting, verification, and agentic workflow) with an assessment design that offers three levels of AI availability. The framework runs in an online assessment platform we built and operate, and it reports interpretable results for each competency backed by evidence, though it is in early development and not yet widely used or validated. We describe the framework and its rationale while outlining a planned evaluation.

CCS Concepts

• **Social and professional topics** → **Student assessment**; *Computer science education*; • **Software and its engineering** → *Designing software.*

Keywords

computing education, AI-assisted programming, competency assessment, student assessment, AI coding assistants

ACM Reference Format:
Eli Young, Andy Luu, and Michael Wehar. 2026. A Competency Framework for Assessing AI-Assisted Programming Work. In *Proceedings of the 2nd ACM Virtual Global Computing Education Conference V.2 (SIGCSE Virtual 2026), November 12–15, 2026, Virtual Event, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3795860.3846260

*Also with Haverford College.

SIGCSE Virtual 2026, Virtual Event, USA

ACM ISBN 979-8-4007-2491-6/2026/11
https://doi.org/10.1145/3795860.3846260

1 Introduction

With modern AI coding assistants, a student can describe what a program should do and have the model produce, run, and fix it, writing little of the code by hand [3, 7, 21]. Students have adopted these tools almost as rapidly as they have grown: in one introductory course, 84 percent had used ChatGPT on their assignments [22], and across 59 computing courses, the share of students willing to submit an AI solution without understanding it rose from 1 to 11 percent in a single year [9].

Teaching students to build with AI creates an assessment problem [1, 14, 19]. A correct-looking program is no longer evidence of the author's skill. A model can produce code that passes a test suite yet is subtly wrong or insecure. When a benchmark's test suite was expanded to roughly 80 times as many cases, measured pass rates fell by up to 29 percent [12], and across four assistants, including GitHub Copilot and ChatGPT, only 21 percent of generated methods were correct even though 31 percent passed their tests [4]. Developers given an AI assistant also wrote less secure code while believing it was more secure [18]. Beginners may be particularly vulnerable to overreliance on AI, as those who most overestimated their own understanding accepted AI suggestions more often and developed an "illusion of competence" [2, 20, 23]. Another study found that beginners did not even run AI-generated code about 40 percent of the time [8]. An instructor cannot separate genuine competence from confident delegation by a passing artifact alone. Therefore, the assessment has to look at the work behind it, including how a student specifies a task, directs an agent, and checks what comes back [5]. We present a competency framework for assessing that work in software development courses, from introductory programming to web development. We also present an assessment design that varies how much AI a student may use. The framework is deployed in an online assessment platform we built and operate.[1]

2 Five Competencies for AI-Assisted Work

Skilled and unskilled programmers work with AI differently, both in what they build and in how they direct the tools. Therefore, even when AI is in the loop, prior-learned programming skills leave observable traces that a proper assessment can measure. The framework scores an AI-assisted session along five competencies based on how skilled programmers work with agents [13, 16], broken down in Table 1 by the observable behaviors that distinguish a high score from a low one. Prior works helped lay the foundation for this framework. These works include Long and Magerko's AI literacy competencies for critically evaluating, collaborating with,

[1] AlgoArena, https://algoarena.net.

Table 1: What constitutes a high or low score for each competency

Competency	Observable behaviors
Problem solving	**High:** names requirements, edge cases, and constraints, and delivers a working, well-scoped result. **Low:** pastes the task in with no further thought.
Planning	**High:** plans before building, asks the model for clarification when needed, and course-corrects when necessary. **Low:** sends a single, unstructured prompt.
Prompting	**High:** writes and refines specific, deliberate prompts. **Low:** repeats vague one-line requests with minimal effort.
Verification	**High:** runs test suites, double-checks that the output or UI is as desired, and asks the model to explain itself when needed. **Low:** applies AI-generated changes blindly.
Agentic workflow	**High:** orchestrates and steers agents in parallel. **Low:** sticks to a single agent and abandons it when it fails.

Table 2: Three levels of AI availability and what each isolates

Assessment	AI available	What it isolates
Agentic	Full agentic tools	Direction and verification
Guided	Restricted assistant that explains or suggests	Problem solving with limited AI assistance
No-AI	None	Baseline ability without AI

and using AI [13], and the trajectory that Mozannar et al. observed programmers move through when working with AI assistants [15]. Recent related work includes scoring students' chatbot conversations in writing courses by hand (DRIVE) [16], defining different levels of permitted AI use (AIAS) [17], and recording and analyzing programming sessions (RECAP) [6]. To our knowledge, however, no prior framework combines defining competency scores from a session recording with linking those scores to supporting evidence while also covering different levels of AI availability.

3 AI Availability and Interpretable Scoring

AI involvement is set per question, from a full agentic workspace to no AI assistance allowed (Table 2). An *agentic* assessment gives the student a full set of AI tools and evaluates how well they direct LLMs and verify their output. Students can set their permission levels (*Careful*, *Balanced*, or *Full access*), and questions can feature real-world developer tools like Claude Code or Codex, running on a virtual machine through a browser-hosted terminal. A *guided* assessment, however, offers a limited assistant that can explain and suggest, but cannot write code directly. A *no-AI* assessment (self-explanatory) neither enables nor scores the process of prompting LLMs, and as a result, the remaining competencies are scored using telemetry from the student's own, unassisted process.

Scoring is automated: behaviors derived from the recorded session [15] (e.g., prompt quality) are translated through a rubric into an interpretable score from 0 to 100 for each competency (Table 1). The session as a whole also receives one of four categories (*Weak*, *Promising*, *Solid*, or *Strong*). Every competency score comes with evidence [11], such as test results, conversation transcripts, and recorded IDE activity, including changes in AI execution mode or movement between questions, so that an instructor can audit conclusions drawn from the assessment.

These signals address the problems raised in Section 1. Problem-solving scores result from the deliverable itself, including its code quality. Verification scores come from whether the student reviewed and tested the output, so that a build that passes its tests but is, for example, unscalable is not taken at face value. Overreliance and the "illusion of competence" are illustrated in Figure 1. Scores also aggregate across a whole class, so an instructor could see where everyone might need improvement.

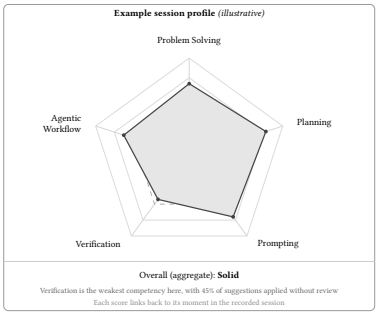

Figure 1: Example session results across the five competencies, scored from 0 to 100.

4 Limitations and Next Steps

The framework has not yet been validated, despite our own informal, encouraging use. In particular, we do not have firm evidence that its ratings accurately reflect a student's competence in AI-assisted programming, so we currently draw no consequential conclusions from the scores. The framework also complements rather than replaces a course's direct assessment of concepts such as graphs, parsing, or garbage collection. Assessing how a student uses AI also presents a challenge where the assessment must avoid rewarding a student's confident delegation over their understanding to catch mistakes [2, 20]. Recording sessions could raise privacy concerns, so if we do collect data for further analysis, students should be told what is logged and why, recordings should be deleted once no longer needed, and further studies should run under informed consent and ethics review.

A next step would be to conduct a formal study with consenting undergraduates in courses piloting the platform. Each participant would complete tasks at three increasing levels of AI availability. Two graders, blind to the framework's grading, would rate every session on the five competencies, and we would compare their ratings with the framework's scores. Task performance would come from the platform's usual test-based grading alongside the graders' code review. This leads us to the following questions. **RQ1:** Do the five competencies capture differences in AI-assisted work that experts agree on? **RQ2:** Do the signals for each competency predict later task performance? Answers to these questions would help instructors adapting to AI [10, 24] address whether assessing a student's collaboration with AI could be a meaningful factor to consider when grading.

Acknowledgments

Portions of this manuscript were drafted and edited with the assistance of AI tools (e.g., Anthropic's Claude). All content was reviewed and verified by the authors.

References

[1] 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 (SIGCSE 2023)*. ACM, New York, NY, USA, 500–506. doi:10.1145/3545945.3569759

[2] Zana Buçinca, Maja Barbara Malaya, and Krzysztof Z. Gajos. 2021. To Trust or to Think: Cognitive Forcing Functions Can Reduce Overreliance on AI in AI-Assisted Decision-Making. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW1, Article 188 (2021). doi:10.1145/3449287

[3] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, et al. 2021. Evaluating Large Language Models Trained on Code. arXiv:2107.03374. doi:10.48550/arXiv.2107.03374

[4] Vincenzo Corso, Leonardo Mariani, Daniela Micucci, and Oliviero Riganelli. 2024. Assessing AI-Based Code Assistants in Method Generation Tasks. In *Proceedings of the 46th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion '24)*. ACM, New York, NY, USA. doi:10.1145/3639478.3643122

[5] Paul Denny, Juho Leinonen, James Prather, Andrew Luxton-Reilly, Thezyrie Amarouche, Brett A. Becker, and Brent N. Reeves. 2024. Prompt Problems: A New Programming Exercise for the Generative AI Era. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024)*. ACM, New York, NY, USA, 296–302. doi:10.1145/3626252.3630909

[6] Keyu He, Qianou Ma, Valerie Chen, Wayne Chi, and Tongshuang Wu. 2026. RECAP: An End-to-End Platform for Capturing, Replaying, and Analyzing AI-Assisted Programming Interactions. arXiv:2605.01104 [cs.SE] doi:10.48550/arXiv.2605.01104

[7] Andrej Karpathy. 2025. There's a new kind of coding I call "vibe coding". X (formerly Twitter). Post, 2 Feb. 2025. https://x.com/karpathy/status/1886192184808149383

[8] Majeed Kazemitabaar, Xinying Hou, Austin Z. Henley, Barbara J. Ericson, David Weintrop, and Tovi Grossman. 2023. How Novices Use LLM-based Code Generators to Solve CS1 Coding Tasks in a Self-Paced Learning Environment. In *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research (Koli Calling '23)*. ACM, New York, NY, USA. doi:10.1145/3631802.3631806

[9] Hieke Keuning, Isaac Alpizar-Chacon, Ioanna Lykourentzou, Lauren Beehler, Christian Köppe, Imke de Jong, and Sergey Sosnovsky. 2024. Students' Perceptions and Use of Generative AI Tools for Programming Across Different Computing Courses. In *Proceedings of the 24th Koli Calling International Conference on Computing Education Research (Koli Calling '24)*. ACM, New York, NY, USA. doi:10.1145/3699538.3699546

[10] 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. In *Proceedings of the 2023 ACM Conference on International Computing Education Research (ICER '23), Volume 1*. ACM, New York, NY, USA. doi:10.1145/3568813.3600138

[11] Raymond Lister, Beth Simon, Errol Thompson, Jacqueline L. Whalley, and Christine Prasad. 2006. Not Seeing the Forest for the Trees: Novice Programmers and the SOLO Taxonomy. In *Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE '06)*. ACM, New York, NY, USA, 118–122. doi:10.1145/1140123.1140157

[12] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*. arXiv:2305.01210. doi:10.48550/arXiv.2305.01210

[13] 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 (CHI '20)*. ACM, New York, NY, USA, 1–16. doi:10.1145/3313831.3376727

[14] Stephen MacNeil, Scott Spurlock, and Ian Applebaum. 2024. Imagining Computing Education Assessment after Generative AI. arXiv:2401.04601 [cs.HC] doi:10.48550/arXiv.2401.04601

[15] Hussein Mozannar, Gagan Bansal, Adam Fourney, and Eric Horvitz. 2024. Reading Between the Lines: Modeling User Behavior and Costs in AI-Assisted Programming. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '24)*. ACM, New York, NY, USA, Article 142. doi:10.1145/3613904.3641936

[16] Manuel Oliveira, Carlos Zednik, Gunter Bombaerts, Bert Sadowski, and Rianne Conijn. 2025. Assessing Students' DRIVE: A Framework to Evaluate Learning through Interactions with Generative AI. *Computers and Education: Artificial Intelligence* 9, Article 100497 (2025). doi:10.1016/j.caeai.2025.100497

[17] Mike Perkins, Leon Furze, Jasper Roe, and Jason MacVaugh. 2024. The Artificial Intelligence Assessment Scale (AIAS): A Framework for Ethical Integration of Generative AI in Educational Assessment. *Journal of University Teaching and Learning Practice* 21, 6 (2024). doi:10.53761/q3azde36

[18] Neil Perry, Megha Srivastava, Deepak Kumar, and Dan Boneh. 2023. Do Users Write More Insecure Code with AI Assistants?. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23)*. ACM, New York, NY, USA. doi:10.1145/3576915.3623157

[19] James Prather, Paul Denny, Juho Leinonen, Brett A. Becker, Ibrahim Albluwi, Michelle Craig, Hieke Keuning, Natalie Kiesler, Tobias Kohn, Andrew Luxton-Reilly, Stephen MacNeil, Andrew Petersen, Raymond Pettit, Brent N. Reeves, and Jaromir Savelka. 2023. The Robots Are Here: Navigating the Generative AI Revolution in Computing Education. In *Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education (ITiCSE-WGR '23)*. ACM, New York, NY, USA. doi:10.1145/3623762.3633499

[20] 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 (ICER '24), Volume 1*. ACM, New York, NY, USA. doi:10.1145/3632620.3671116

[21] Advait Sarkar and Ian Drosos. 2025. Vibe Coding: Programming through Conversation with Artificial Intelligence. In *Proceedings of the 36th Annual Conference of the Psychology of Programming Interest Group (PPIG 2025)*. arXiv:2506.23253. doi:10.48550/arXiv.2506.23253

[22] Andreas Scholl and Natalie Kiesler. 2024. How Novice Programmers Use and Experience ChatGPT when Solving Programming Exercises in an Introductory Course. In *2024 IEEE Frontiers in Education Conference (FIE)*. IEEE. doi:10.1109/FIE61694.2024.10893442

[23] 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 *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems (CHI EA '22)*. ACM, New York, NY, USA. doi:10.1145/3491101.3519665

[24] Cynthia Zastudil, Magdalena Rogalska, Christine Kapp, Jennifer Vaughn, and Stephen MacNeil. 2023. Generative AI in Computing Education: Perspectives of Students and Instructors. In *2023 IEEE Frontiers in Education Conference (FIE)*. IEEE. doi:10.1109/FIE58773.2023.10343467