

How Is AI Reshaping Computer Science Students' Cognition of Programming?

Ruolin Li

University of Limerick, Ireland

li.ruolin@ul.ie

Jim Buckley

University of Limerick, Ireland

Jim.Buckley@ul.ie

Tabea De Wille

University of Limerick, Ireland

Tabea.Dewille@ul.ie

Abstract

Generative AI is increasingly embedded in computer science education, creating new opportunities and concerns regarding how students learn to program. This research investigates AI's impact on computer science students' cognition by examining how cognitive constructs are represented across programming-education research and how their significance varies across software engineering activities in AI and non-AI literature. A systematic mapping study, informed by established software engineering mapping guidelines and PRISMA 2020, screened 14,628 records between 2023 and 2025 from six databases and retained 191 primary studies, including 80 AI-related and 111 non-AI studies. Mixed deductive-inductive thematic analysis identified twelve cognitive and pedagogical themes. Cognitive processes, computational thinking, and comprehension were most frequently examined, while metacognition, motivation, cognitive load, misconceptions, and mental models capture complementary regulatory and interpretive dimensions. Findings indicate that cognition is task-sensitive: debugging foregrounds program comprehension, misconception diagnosis, and evaluation, whereas collaborative and pair-programming activities foreground explanation, shared problem solving, and cognitive regulation. AI-related studies predominantly address reasoning, interaction, and regulation, while foundational constructs such as comprehension, misconceptions, and mental models remain more prominent in non-AI research. Rather than viewing this distribution solely as conceptual fragmentation, this research proposes that it reflects both task specialisation and the early development of AI-mediated programming education. The mapping provides evidence that programming cognition is task-sensitive and identifies a research gap between AI-assisted code production and the verification, diagnosis, and judgement activities that remain central to durable learning. This gap motivates longitudinal research on AI amplification, cognitive debt, and emerging mental labour.

Keywords: generative AI; programming education; computer science education; cognition; systematic mapping study; computational thinking; metacognition

1. Background and research gap

The rapid adoption of generative artificial intelligence (GenAI) in the digital world has fundamentally changed how programming is learned and practiced (Becker et al., 2022). In industry settings, AI-assisted tools have demonstrated substantial productivity gains among developers (Peng et al., 2023; Paradis et al., 2025), calling for the acquisition of a different skillset and curriculum change in the third-level education of software engineering students. In parallel, educators increasingly integrate AI into programming instruction, feedback, and assessment practices, while students exhibit diverse proactive and reactive behaviours as they interact with AI throughout their learning journeys (Prather et al., 2023; Liu et al., 2024; Sheese et al., 2024; Güner and Er, 2025). As a result, computer science students are no longer learning programming under the same cognitive conditions as previous generations.

Large language model tools can generate code, explain snippets, propose tests, and participate in debugging dialogues. These capabilities may support experimentation and immediate feedback, but they also make it possible to complete programming tasks without establishing robust conceptual understanding (Prather et al., 2024). The educational question is therefore not only whether students use AI, but how AI changes the reasoning, evaluation, self-regulation, and knowledge construction through which programming is learned, and if those changes leave graduates in a better or worse position for AI-supported software engineering careers.

This question sits alongside a longer tradition of research on programming cognition. Computer science education has examined computational thinking, program comprehension, misconceptions,

debugging, mental models, cognitive load, and metacognition as related but distinct dimensions of programming learning (Wing, 2006; Soloway and Ehrlich, 1984; Pennington, 1987; Qian and Lehman, 2018; McCauley et al., 2008; Sweller, 1988; Prather et al., 2018). Recent reviews have considered GenAI adoption, assessment, and pedagogical practice in computing education (Prather et al., 2023; Golrang and Sharma, 2025), while other reviews concentrate on particular cognitive constructs such as computational thinking or cognitive load (Melro et al., 2023; Suryani et al., 2024). However, these strands rarely provide an integrated account of where cognition is studied, in which programming tasks, and how AI-related research compares with the adjacent non-AI literature.

Like Mansoor et al. (2024), we argue that such a task-sensitive perspective matters. Programming cognition is enacted across activities including code generation, debugging, program comprehension, testing, review, and collaboration, rather than solely during code writing; these activities are also shaped by the presence or absence of AI support. Generative AI may redistribute, rather than eliminate, cognitive effort: while it can accelerate code production, learners must still inspect, test, diagnose, judge, and integrate generated outputs (Robins, Rountree and Rountree, 2003; Bacchelli and Bird, 2013; Vaithilingam, Zhang and Glassman, 2022; Barke, James and Polikarpova, 2023), particularly when those outputs appear plausible but are incorrect (Pearce et al., 2022). The mapping therefore regards software engineering tasks as the contexts in which cognitive constructs acquire educational meaning.

2. Research questions

RQ1a. What cognitive and pedagogical themes are addressed in programming-education research?

RQ1b. In which software engineering task contexts do the proposed themes occur?

RQ2a. How is the mapped literature distributed between AI-related and non-AI studies?

RQ2b. What thematic relationships are visible across the mapped literature?

RQ3. What research gaps emerge from the interaction between programming cognition, software engineering tasks and AI assistance?

3. Mapping study completed to date

A systematic mapping study synthesised 191 primary studies: 80 substantively examine GenAI in programming-learning contexts and 111 provide the analogous non-AI knowledge base needed to interpret trends concurrent with AI adoption. A deductive-inductive coding scheme captured 12 themes and the programming tasks in which they appeared. Multiple codes were permitted for each article. The primary author completed the coding; the second and final authors each reviewed 12 articles, and differences were resolved through debate towards consensus. Article-level co-occurrence analysis served as an internal analytic cross-check of the thematic map, rather than an independent methodological triangulation.

4. Findings and conceptual synthesis

Together, RQ1a and RQ1b show that programming cognition is not a single phenomenon. As shown in figure 1, the literature is cognitively centred, led by cognitive processes ($n = 82$), computational thinking ($n = 51$), and comprehension ($n = 38$), but its emphasis changes with the task. Programming projects are the most frequent task family ($n = 51$), followed by debugging ($n = 36$), function writing ($n = 25$), program comprehension ($n = 24$), and collaborative development ($n = 19$). Debugging makes comprehension, misconception diagnosis, cognitive effort, and evaluation conspicuous; collaborative development foregrounds explanation, help-seeking, and shared regulation.

The answer to RQ2(a/b) shows a literature in transition. AI-related studies cluster around cognitive processes, interaction, and learning regulation, while comprehension, misconceptions, and mental models remain areas of interest even after the emergence of AI. Across the coded corpus, 41 of 66 possible undirected theme pairs occurred. The most common pairing is comprehension-misconceptions ($n = 17$), followed by cognitive processes-comprehension ($n = 10$) and cognitive processes-computational thinking ($n = 10$). Figure 1 presents AI as a cross-cutting influence on an established cognitive-pedagogical landscape, not as a force that replaces its foundations.

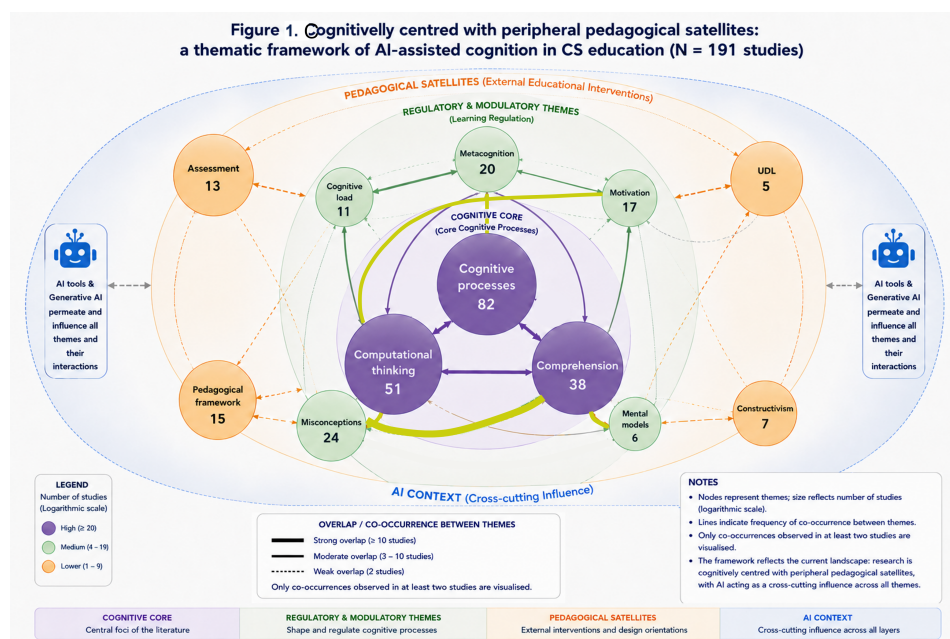


Figure 1 - Conceptual synthesis derived from thematic frequencies and article-level co-occurrence analysis. Node size reflects the number of studies; line thickness reflects observed co-occurrence frequency.

RQ3 is therefore framed as an evidence-led gap. Figure 2 shows that AI-related studies are concentrated in programming projects (31 AI-related versus 20 non-AI studies) and code completion (9 versus 5). In contrast, code comprehension (5 versus 18), debugging (15 versus 21), and collaborative/pair programming (7 versus 12) receive comparatively less AI-related coverage; code review is balanced (3 versus 3), and testing is nearly absent. The mapping cannot show that AI has already redistributed student effort and cognitive load as we expected, yet it does show that research coverage has not yet fully followed the verification, diagnosis, review, and testing work that AI-assisted programming makes educationally consequential.

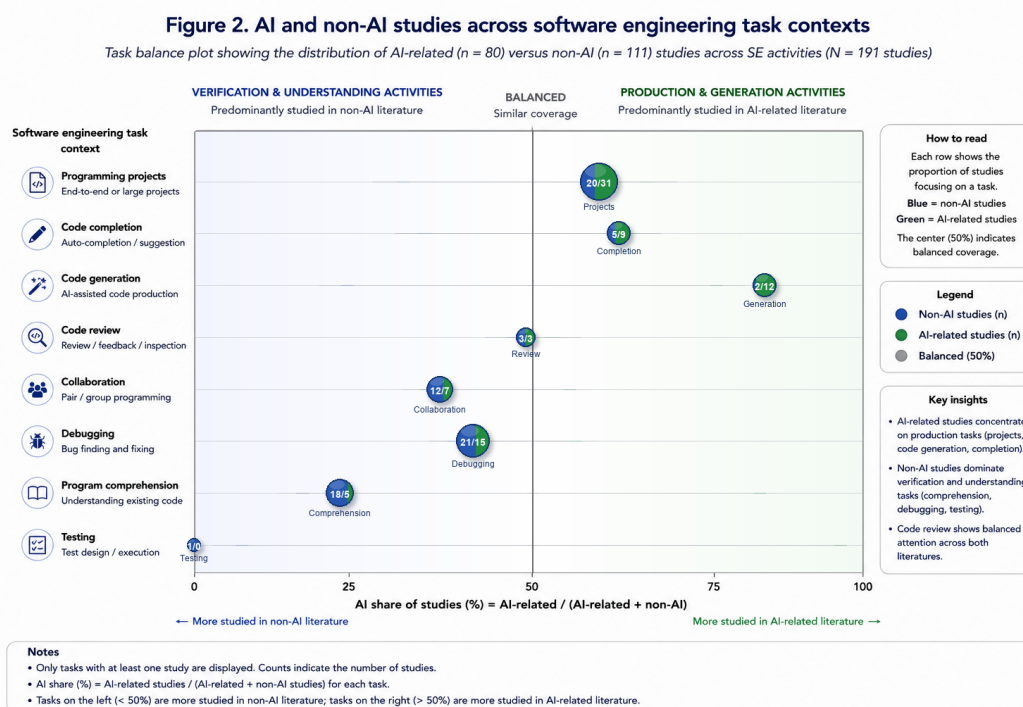


Figure 2 - AI-related and non-AI studies by software engineering task. N = 191 task-coded primary studies; multiple task coding was permitted.

5. Doctoral research agenda

The next phase of the doctorate will develop integrated, task-sensitive accounts of learning with AI by specifying the programming task, form of AI assistance, learner role, intended cognitive process, pedagogy, and assessment evidence. Two linked research directions are central. First, AI amplification and individual cognitive trajectories will examine how prior knowledge, self-regulation, confidence, motivation, accessibility needs, and help-seeking shape AI use over time. The working proposition is that AI may amplify learner tendencies: reflective students may test and compare suggestions, while anxious, avoidant, or less experienced students may accept the first plausible answer prematurely. We hypothesize that this pattern may exacerbate differences in high-achievers and poorer students.

Second, the project will investigate cognitive load, cognitive debt, and emerging mental labour. Cognitive debt describes cases in which a learner completes an AI-assisted task while accumulating incomplete understanding that later constrains retention, debugging, code review, and system-level reasoning. Longitudinal, mixed-method research is needed to determine when AI frees cognitive capacity for learning and when it transfers effort toward verification, comparison, orchestration, and responsibility for judging outputs. This agenda reframes executable output as necessary but insufficient evidence of learning; future measures should also capture explanation, error diagnosis, misconception repair, metacognitive monitoring, independent performance after assistance is withdrawn, and transfer to new programming problems.

References

- Bacchelli, A. and Bird, C. (2013) 'Expectations, outcomes, and challenges of modern code review', Proceedings of the 35th International Conference on Software Engineering, pp. 712–721. doi: 10.1109/ICSE.2013.6606617.
- Barke, S., James, M.B. and Polikarpova, N. (2023) 'Grounded Copilot: How programmers interact with code-generating models', Proceedings of the ACM on Programming Languages, 7(OOPSLA1), pp. 85–111. doi: 10.1145/3586030.
- Becker, B.A., Denny, P., Finnie-Ansley, J., Luxton-Reilly, A., Prather, J., and Santos, E.A. (2022) 'Programming Is Hard – Or at Least It Used to Be: Educational Opportunities And Challenges of AI Code Generation', doi: 10.48550/arxiv.2212.01020.
- Golrang, A. and Sharma, K. (2025) 'The role of AI in shaping educational experiences in computer science: A systematic review', Computers in Human Behavior: Artificial Humans, 6, doi: 10.1016/j.chbah.2025.100199.
- Güner, H. and Er, E. (2025) 'AI in the classroom: Exploring students' interaction with ChatGPT in programming learning', Education and information technologies, 30(9), 12681–12707, doi: 10.1007/s10639-025-13337-7.
- Liu, R., Zenke, C., Liu, C., Holmes, A., Thornton, P., and Malan, D.J. (2024) 'Teaching CS50 with AI: Leveraging Generative Artificial Intelligence in Computer Science Education', in Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1, New York, NY, USA: ACM, 750–756, doi: 10.1145/3626252.3630938.
- Mansoor, N., Peterson, C.S., Dodd, M.D., and Sharif, B. (2024) 'Assessing the Effect of Programming Language and Task Type on Eye Movements of Computer Science Students', ACM transactions on computing education, 24(1), doi: 10.1145/3632530.
- McCauley, R., Fitzgerald, S., Lewandowski, G., Murphy, L., Simon, B., Thomas, L., and Zander, C. (2008) 'Debugging: a review of the literature from an educational perspective', Computer science education, 18(2), 67–92, doi: 10.1080/08993400802114581.
- Melro, A., Tarling, G., Fujita, T., and Kleine Staarman, J. (2023) 'What Else Can Be Learned When Coding? A Configurative Literature Review of Learning Opportunities Through Computational

- Thinking', *Journal of educational computing research*, 61(4), 901–924, doi: 10.1177/07356331221133822.
- Paradis, E., Grey, K., Madison, Q., Nam, D., Macvean, A., Meimand, V., Zhang, N., Ferrari-Church, B., and Chandra, S. (2025) 'How Much Does AI Impact Development Speed? an Enterprise-Based Randomized Controlled Trial', in *IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice (Online)*, IEEE, 618–629, doi: 10.1109/ICSE-SEIP66354.2025.00060.
- Pearce, H., Ahmad, B., Tan, B., Dolan-Gavitt, B. and Karri, R. (2022) 'Asleep at the Keyboard? Assessing the security of GitHub Copilot's code contributions', *2022 IEEE Symposium on Security and Privacy*, pp. 754–768. doi: 10.1109/SP46214.2022.9833571.
- Peng, S., Kalliamvakou, E., Cihon, P., and Demirer, M. (2023) 'The Impact of AI on Developer Productivity: Evidence from GitHub Copilot', doi: 10.48550/arxiv.2302.06590.
- Pennington, N. (1987) Stimulus structures and mental representations in expert comprehension of computer programs. *Cognitive Psychology* 19, 295–341.
- Prather, J., Denny, P., Leinonen, J., Becker, B.A., Albluwi, I., Craig, M., Keuning, H., Kiesler, N., Kohn, T., Luxton-Reilly, A., MacNeil, S., Petersen, A., Pettit, R., Reeves, B.N., and Savelka, J. (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*, New York, NY, USA: ACM, 108–159, doi: 10.1145/3623762.3633499.
- Prather, J., Pettit, R., McMurtry, K., Peters, A., Homer, J., and Cohen, M. (2018) 'Metacognitive Difficulties Faced by Novice Programmers in Automated Assessment Tools', in *Proceedings of the 2018 ACM Conference on International Computing Education Research*, New York, NY, USA: ACM, 41–50, doi: 10.1145/3230977.3230981.
- Prather, J., Reeves, B., Leinonen, J., MacNeil, S., Randrianasolo, A.S., Becker, B., Kimmel, B., Wright, J., and Briggs, B. (2024) 'The Widening Gap: The Benefits and Harms of Generative AI for Novice Programmers', doi: 10.48550/arxiv.2405.17739.
- Qian, Y. and Lehman, J. (2018) 'Students' Misconceptions and other difficulties in introductory programming: A literature review: A Literature Review', *ACM transactions on computing education*, 18(1), doi: 10.1145/3077618.
- Robins, A., Rountree, J. and Rountree, N. (2003) 'Learning and teaching programming: A review and discussion', *Computer Science Education*, 13(2), pp. 137–172. doi: 10.1076/csed.13.2.137.14200.
- Sheese, B., Liffiton, M., Savelka, J., and Denny, P. (2024) 'Patterns of Student Help-Seeking When Using a Large Language Model-Powered Programming Assistant', in Seton, C. and Herbert, N., eds, *Proceedings of the 26th Australasian Computing Education Conference*, New York, NY, USA: ACM, 49–57, doi: 10.1145/3636243.3636249.
- Soloway, E. and Ehrlich, K. (1984) 'Empirical studies of programming knowledge', *IEEE Transactions on Software Engineering*, SE-10(5), pp. 595–609. doi: 10.1109/TSE.1984.5010283.
- Suryani, M., Santoso, H.B., Schrepp, M., Aji, R.F., Hadi, S., Sensuse, D.I., Suryono, R.R., and Kautsarina (2024) 'Role, Methodology, and Measurement of Cognitive Load in Computer Science and Information Systems Research', *IEEE access*, 12, 190007–190024, doi: 10.1109/ACCESS.2024.3514355.
- Sweller, J. (1988) 'Cognitive load during problem solving: Effects on learning', *Cognitive science*, 12(2), 257–285, doi: 10.1207/s15516709cog1202_4.
- Vaithilingam, P., Zhang, T. and Glassman, E.L. (2022) 'Expectation vs. Experience: Evaluating the usability of code generation tools powered by large language models', *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, pp. 1–7. doi: 10.1145/3491101.3519665.
- Wing, J.M. (2006) 'Computational thinking', *Communications of the ACM*, 49(3), pp. 33–35. doi: 10.1145/1118178.1118215.