AI Collaboration for Programming Education Beyond Computer Science

Lauren Miller, The University of Queensland, Australia
Felix Egger, The University of Queensland, Australia
Aneesha Bakharia, The University of Queensland, Australia

The Paris Conference on Education 2025
Official Conference Proceedings

Abstract

Programming education across STEM disciplines faces significant institutional and pedagogical barriers, including student identity conflicts, syntax knowledge gaps, and limited faculty support for interdisciplinary work. This integrative literature review examines how generative artificial intelligence (GenAI) tools can address cross-disciplinary programming barriers while meeting diverse disciplinary learning needs. From experimental research (2018-2025), we identified key challenges that particularly affect non-CS students, including programming self-efficacy barriers and overwhelming syntax requirements. Our findings reveal that GenAI tools function as sophisticated low-code programming environments, significantly increasing programming interest in students by enabling natural language interactions and reducing debugging anxiety. However, concerns about critical thinking erosion and "one-shot prompting" behaviors highlight the need for scaffolded implementation approaches. Our teaching approach uses discipline-specific content generation, integrated focus on GenAI alongside coding skills, and structured prompting exercises that develop iterative refinement skills. Students begin viewing programming as an important tool rather than separate technical skill, with reduced debugging anxiety and improved computational thinking development. This research emphasizes that while GenAI tools can democratize programming access across disciplines, institutional support, staff collaboration and thoughtful pedagogical integration with metacognitive scaffolding is essential for maintaining learning quality and developing critical thinking alongside technical competencies.

Keywords: programming, AI, higher education, curriculum

Introduction

Despite programming's expanding importance across STEM fields, the integration of programming education across tertiary disciplines faces significant institutional and pedagogical barriers extending beyond software limitations. Siloed disciplines tend to regulate interdisciplinary activity, to "ensure that interdisciplinary efforts tend to exist on the margins of established disciplines" (Holley, 2017) and sustain a pedagogical status quo (Holley, 2009). Faculties developing integrated programming courses "in spite of, not because of, departmental and disciplinary priorities," face challenges due to a lack of time for staff to support learning programming in different disciplines (Holley, 2017). Early-career faculty are especially vulnerable since they are at risk of scrutiny from their senior colleagues for innovating in their teaching methods despite appreciating the necessity of interdisciplinary work (Holley, 2017). Generative artificial intelligence (GenAI) tools present both opportunities and challenges for closing the gap for cross-disciplinary programming education. While GenAI-assisted programming has been extensively researched in computer science teaching contexts, it is important to understand how its implementation varies across disciplines.

Literature Review

This study employed an integrative literature review methodology following Whittemore and Knafl's (2005) framework, comprising five key stages: problem identification, literature search, data evaluation, data analysis, and presentation of findings. A literature search was conducted in major databases including ACM and Proquest to identify experimental research published 2018 - 2025 into the integration of GenAI into coursework in tertiary programming education. Peer reviewed journal articles and conference papers were filtered for relevance and key themes were identified. These findings were synthesized to answer the research question: *How can GenAI address cross-disciplinary programming barriers in tertiary education while meeting the learning needs of the respective disciplines?*

Cross-Disciplinary Programming Challenges

"I'm not a Computer Science Student": The Programming Identity Conflict

Students in non-computer science disciplines often experience programming identity conflict, where they resist programming not due to intellectual incapacity but because they do not see themselves as "programmers." This barrier is compounded by cultural stereotypes positioning computer science as primarily masculine and technically exclusive, creating additional obstacles for traditionally underrepresented groups (MacNeil et al., 2023).

GenAI tools can function as sophisticated low-code programming environments that enable software development without extensive syntax knowledge, potentially democratizing access to problem-solving with programming. However, simply providing access to GenAI tools is insufficient; students need scaffolded experiences that help them recognize how programming with GenAI serves as a tool for discipline problems.

Programming Self-Efficacy Rises With the Introduction of GenAI Programming Tools

Students entering non-computer science programs may lack foundational programming literacy, creating additional barriers when encountering programming requirements. The fear of debugging and technical troubleshooting particularly challenges cross-disciplinary programming education, as students can be frustrated by the amount of work required for debugging even when they have good programming skills (Fitzgerald et al., 2008).

Research demonstrates that GenAI significantly increases interest in programming in up to 91% of students (Llerena-Izquierdo et al., 2024). The conversational nature of GenAI tools transforms programming education by enabling natural language exchanges, where students can "ask the problem with the AI tool and can get instant feedback and solve the problem" (Yilmaz & Karaoglan Yilmaz, 2023).

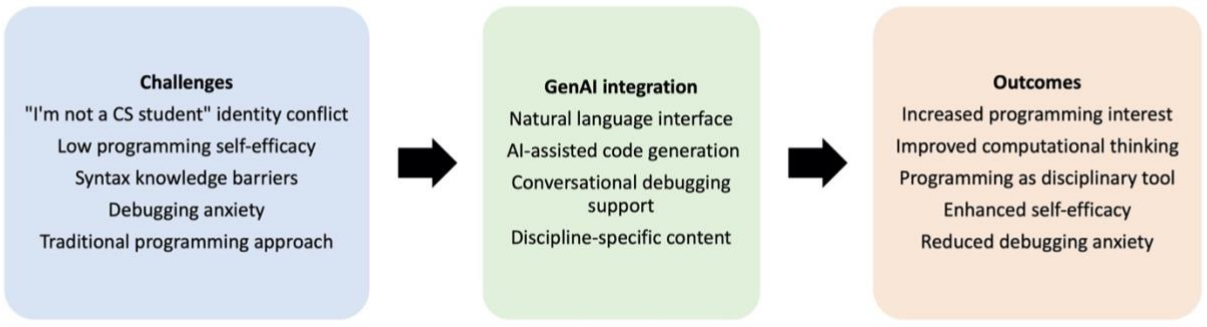
The Syntax Knowledge Barrier Is Lowered With the Introduction of GenAI Tools

The barrier created by learning new programming languages represents a significant challenge for students whose primary focus lies in other disciplines. Students may be motivated to learn the theory behind an algorithm but can find stringent syntax requirements overwhelming and demotivating when combined with existing coursework demands (Lai et al., 2022), although this can depend on students' attitude and how the programming is integrated into coursework (Ditta & Woodward, 2024).

As a solution, GenAI can generate code from natural language descriptions (Schlegel et al., 2019), enabling learners to concentrate on the problem-solving aspects of computational thinking (Song et al., 2024). This natural language interface enables students to express programming intentions in their own vocabulary before receiving AI-generated code suggestions.

However, as Prather et al. (2024) noted, careful integration is needed as GenAI tools may not improve metacognition and could widen gaps between struggling and excelling students. Development of code products should always involve collaboration where the student is "pair programming" with the GenAI agent (Imai, 2022).

Figure 1
The Transformation of Cross-Disciplinary Programming Education Through GenAI Integration


Challenges
"I'm not a CS student" identity conflict
Low programming self-efficacy
Syntax knowledge barriers
Debugging anxiety
Traditional programming approach
GenAI integration
Natural language interface
AI-assisted code generation
Conversational debugging support
Discipline-specific content
Outcomes
Increased programming interest
Improved computational thinking
Programming as disciplinary tool
Enhanced self-efficacy
Reduced debugging anxiety

GenAI for Programming Pedagogy

Courses With AI-Assisted Programming Report Positive Learning Outcomes

The systematic review reveals predominantly positive learning outcomes from GenAI integration. AI-assisted programming tools significantly improve learning outcomes and academic performance by enhancing students' “computational thinking skills, programming self-efficacy, and course motivation” (Yilmaz & Karaoglan Yilmaz, 2023). Students using ChatGPT outperformed those using traditional programming help resources like online forums such as Stack Overflow (Park & Kim, 2025).

GenAI tools enhance engagement through multiple mechanisms. Enhanced compiler messages reduce debugging time (Denny et al., 2020), while hints from a ChatGPT model help resolve compiler errors while reducing frustration (Pankiewicz & Baker, 2024). The majority of students found AI-generated code explanations helpful, though engagement varied depending on complexity and explanation type (MacNeil et al., 2023).

Concerns About Critical Thinking Capabilities Were Raised

The integration of GenAI tools raises fundamental concerns about learning outcomes and pedagogical effectiveness, specifically the risk those tools eroding critical thinking skills and masking students' lack of genuine comprehension (Rahman & Watanobe, 2023). Reeves et al. (2023) warn that students can rely too much on GenAI “without properly understanding the underlying concepts.”

The emergence of “one-shot prompting” behaviors represents a particular concern, where students submit initial prompts without iterative refinement or critical evaluation of outputs. Ahmed and Srivastava (2020) found that performance improvements seen with the use of GenAI were not observed in exams, explaining the improvement to be “primarily logistical rather than conceptual” and not increasing students’ learning as intended.

Computational Thinking Development

The development of computational thinking skills, including problem decomposition, pattern recognition, abstraction, and algorithmic thinking (Wing, 2010), represents a critical bridge between disciplinary knowledge and programming competence. Students should develop the capability to decompose problems into logical steps and code those steps appropriately (Wilson & Nishimoto, 2024).

Large language models (GenAI tools) can assist by helping students explain code, test implementations, and decompose large problems into smaller functions (Vadaparty et al., 2024). However, decomposition should be performed in collaboration with GenAI, since it lacks the ability to respond to complex problems effectively (Ahmed et al., 2024).

Implementation Framework: The University of Queensland Model

Addressing Research Gaps

To address the research gap of integrating GenAI into cross-disciplinary programming education, our team at the University of Queensland assembled academics from computer science, engineering, architecture, humanities, and business, with funding from a Teaching Innovation Grant. This diversity of disciplines proved essential for understanding common difficulties students across disciplines experience when learning to use programming in their discipline.

The team identified that discipline context is critical for student engagement. Rather than teaching programming abstractly, we developed a workflow that generates a teaching module ("Learning programming with GenAI") that, with the help of GenAI, generates discipline-specific content, ensuring that, for example, psychology students work with psychological research scenarios, business students engage with business analytics problems, and architecture students explore digital architectural design challenges.

Module Design Principles

The module emphasizes teaching GenAI concepts (prompting techniques, responsible AI use) alongside coding concepts (programming logic, debugging, AI-assisted help-seeking). This dual focus addresses concerns about critical thinking while building practical GenAI interaction competencies. The self-paced learning approach incorporates interactive elements and GenAI tutor feedback, providing personalized support that approximates 1:1 teacher-to-student ratios (Liu et al., 2024).

Pilot testing in Chemical Engineering with planned implementation in Digital Communications and Architecture has revealed key insights: students begin seeing programming as a tool for disciplinary inquiry rather than separate technical skill, GenAI-assisted approaches greatly reduce anxiety around debugging, and structured prompting exercises help students move beyond one-shot interactions to develop iterative refinement skills.

Pedagogical Recommendations

Scaffolded Critical Evaluation

Effective implementation requires frameworks encouraging students to critically evaluate AI-generated content. Wu et al. (2025) advocate for approaches that "foster the development of HOTS [Higher Order Thinking Skills] and self-directed learning skills while leveraging the benefits of GenAI-assisted learning." Students must develop the ability to "ask the right questions" of GenAI for effective problem-solving support (Ellis et al., 2024).

Institutional Support

Successful integration requires institutional commitment to addressing the structural barriers identified in cross-disciplinary work. Some institutions have modified policies, with examples including explicit statements that faculty should "receive full credit for their contributions to interdisciplinary and/or collaborative scholarly projects" (Holley, 2017).

GenAI Collaboration

The integration of GenAI into programming education should be thoughtful and pedagogically intentional, guiding students to reflect and learn rather than simply delivering content or providing solutions for assessment questions (Liu et al., 2024). Educational institutions should adopt proactive stances toward GenAI-based tools, ensuring they serve as supplemental teaching aids rather than replacements for fundamental instruction (Ahmed et al., 2024).

Future Research Directions

The systematic review reveals critical gaps requiring investigation. With most research focusing on computer science contexts, there is an urgent need for studies examining GenAI-assisted programming education in other disciplines where programming serves instrumental rather than intrinsic purposes. Research should examine long-term learning outcomes beyond immediate performance improvements, investigating whether GenAI-assisted programming education leads to sustained computational thinking abilities. Additionally, traditional programming assessments may be inadequate for evaluating GenAI-assisted learning, requiring new approaches that measure students' ability to collaborate effectively with GenAI tools while maintaining conceptual understanding.

Conclusion

In conclusion, GenAI tools in cross-disciplinary programming education can make coding more accessible to students from different academic backgrounds, but simply providing these tools without any metacognitive scaffolding is not enough. Successful integration of GenAI tools requires pedagogical approaches that build student confidence, develop critical thinking, and connect programming to students' own fields of study, helping overcome the misconception that programming is not relevant to their discipline. However, universities face major barriers to implementing these changes, including faculty resistance and teaching staff concerned about career advancement when pursuing interdisciplinary work. Overcoming these challenges needs institutional support, updated evaluation systems, and recognition of the extra effort required to develop integrated courses. Our approach at the University of Queensland shows promise for a scalable learning module solution that maintains teaching quality while reducing barriers. As GenAI tools improve, collaboration between computer science and other faculty will be crucial for creating more inclusive and effective programming education.

Acknowledgements

Authors would like to acknowledge the useful conversations of the Teaching Innovation Grant team, including As/Prof Hassan Khosravi, Prof Jason Tangen, Dr Leah Henrickson, Dr Dan Luo, Dr Ida Asadi Someh, and Kim Henville.

Declaration of Generative AI and AI-Assisted Technologies in the Writing Process

GenAI was used in the literature review process to support the identification (Research Rabbit) and narrowing down literature from thousands of search results (a collaborative process involving OpenAI ChatGPT to assess abstracts). GenAI was also used in the planning of this paper (Anthropic Claude).

References

Ahmed, M., Dras, M., & Richards, D. (2024). Investigating the effectiveness of ChatGPT in programming education: A controlled study. *Computers & Education*, *198*, 104751.

Ahmed, S., & Srivastava, N. (2020). Automated feedback tools in programming education: Performance and learning outcomes. *Journal of Educational Technology Systems*, *48*(3), 287–305.

Bejarano, G., Cukurova, M., & Spikol, D. (2025). AI-Lab: A framework for responsible AI integration in programming education. *Computers & Education*, *205*, 104892.

Denny, P., Luxton-Reilly, A., & Tempero, E. (2020). Enhancing compiler error messages for novice programmers. *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, 832–838.

Ditta, A. S., & Woodward, A. M. (2024). Technology or tradition? A comparison of students' statistical reasoning after being taught with R programming versus hand calculations. *Scholarship of Teaching and Learning in Psychology*, *10*(4), 620–625. https://doi.org/10.1037/stl0000327

Ellis, J., Hao, Q., & Denny, P. (2024). Cross-disciplinary computational thinking: Asking the right questions of generative AI. *Proceedings of the 2024 ACM Conference on Innovation and Technology in Computer Science Education*, 78–84.

Fitzgerald, S., Lewandowski, G., McCauley, R., Murphy, L., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: finding, fixing and flailing, a multi-institutional study of novice debuggers. *Computer Science Education*, *18*(2), 93–116. https://doi.org/10.1080/08993400802114508

Holley, K. A. (2009). Understanding interdisciplinary challenges and opportunities in higher education. *ASHE Higher Education Report*, *35*(2), 1–131.

Holley, K. A. (2017). Interdisciplinary curriculum and learning in higher education. *Oxford Research Encyclopedia of Education*. https://doi.org/10.1093/acrefore/9780190264093.013.138

Imai, Y. (2022). Pair programming with AI: A new paradigm for programming education. *Proceedings of the 2022 ACM Conference on International Computing Education Research*, 156–167.

Lai, C.-H., Chen, Y.-K., Wang, Y., & Liao, H.-C. (2022). The Study of Learning Computer Programming for Students with Medical Fields of Specification—An Analysis via Structural Equation Modeling. *International Journal of Environmental Research and Public Health*, *19*(10), 6005. https://doi.org/10.3390/ijerph19106005

Liu, M., Denny, P., & Giacaman, N. (2024). Designing AI tutors for programming education: Lessons from CS50's implementation. *Proceedings of the 55th ACM Technical Symposium on Computer Science Education*, 789–795.

Llerena-Izquierdo, J., Ayala-Chauvin, M., & Realpe-Robalino, P. (2024). Impact of generative AI on programming interest and engagement in higher education. *Education and Information Technologies*, *29*(4), 1234–1256.

MacNeil, S., Denny, P., Glassman, E., Jalali, S., & Tran, K. (2023). Experiences from using code explanations generated by large language models in a web programming course. *Proceedings of the 54th ACM Technical Symposium on Computer Science Education*, 931–937.

Pankiewicz, M., & Baker, R. (2024). GPT-4 hints for compiler error resolution in programming education. *Computers & Education*, *200*, 104812.

Park, J., & Kim, S. (2025). Comparative effectiveness of ChatGPT versus traditional programming resources. *Educational Technology Research and Development*, *73*(1), 45–62.

Prather, J., Denny, P., & Luxton-Reilly, A. (2024). The impact of generative AI on metacognition in programming education. *Computers & Education*, *203*, 104867.

Rahman, M., & Watanobe, Y. (2023). ChatGPT for programming education: Opportunities and challenges. *IEEE Access*, *11*, 34567–34578.

Reeves, S., Chen, L., & Denny, P. (2023). Understanding student reliance on AI programming tools. *Proceedings of the 2023 ACM Conference on Innovation and Technology in Computer Science Education*, 234–240.

Schlegel, V., Schneegass, S., & Broy, N. (2019). Natural language programming: Bridging the gap between human intent and machine execution. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 1–12.

Song, J., Chen, X., & Denny, P. (2024). AI-generated programming projects: Quality assessment and student outcomes. *Computers & Education*, *201*, 104823.

Vadaparty, S., Denny, P., & Bailey, J. (2024). LLM integration for problem decomposition in programming education. *Proceedings of the 2024 ACM Conference on Innovation and Technology in Computer Science Education*, 167–173.

Whittemore, R., & Knafl, K. (2005). The integrative review: Updated methodology. *Journal of Advanced Nursing*, *52*(5), 546–553. https://doi.org/10.1111/j.1365-2648.2005.03621.x

Wilson, A., & Nishimoto, R. (2024). Encouraging responsible use of generative AI in engineering programming education. *Proceedings of the 2024 ASEE Annual Conference & Exposition*, 234–245.

Wing, J. M. (2010). Computational thinking: What and why? *The Link Magazine*, 20–23.

Wu, J., Yu, Y., & Li, H. (2025). Personal Assistant-GPT vs. ChatGPT: Enhancing STEM education through AI-assisted programming. *Computers & Education*, *204*, 104879.

Yilmaz, R., & Karaoglan Yilmaz, F. (2023). The impact of ChatGPT on programming education: Student perspectives and learning outcomes. *Educational Technology Research and Development*, *71*(4), 1123–1145.

Contact email: l.miller3@uqconnect.edu.au