

DESIGNING PEDAGOGICAL GUARDRAILS FOR DELIBERATE AI-ASSISTED PROGRAMMING IN FIRST-YEAR COMPUTING

1. INTRODUCTION

Generative AI is becoming increasingly embedded in both software engineering and tertiary education. AI-assisted development tools are reshaping programming practice by shifting some work away from line-by-line coding and towards intention specification, guided revision, and iterative collaboration with AI (Ozkaya, 2023), while students are also encountering tools such as ChatGPT, Val at RMIT, and GitHub Copilot in their everyday study routines.

In first-year computing, however, simple access to these tools is not enough. Meaningful use of generative AI requires judgment, self-regulation, and disciplinary understanding, yet novice students are often left to use such tools with limited guidance. The pedagogical priority in introductory programming is therefore not rapid proficiency in AI, but the development of foundational programming knowledge, computational thinking, and critical analysis. Learning to work skilfully with generative AI matters, but in the first year, this must remain secondary to learning how to think through problems, understand code, and justify solutions.

Novice learners often lack the knowledge to evaluate, validate, or refine AI-generated code effectively. The question for educators is therefore not whether students will use generative AI, but how that use is scaffolded so that it supports rather than prevents learning. This paper presents the early development of a pedagogical framework and workshop intervention designed to support deliberate, accountable AI-assisted programming in first-year computing courses.

2. AI-ASSISTED PROGRAMMING AS A PEDAGOGICAL DESIGN PROBLEM

Emerging work in computing education suggests that novice students often use generative AI in ways that privilege speed and surface functionality over understanding. Studies in this area point to developmentally uneven outcomes, “autocomplete-dependent” programming, vague prompting, and limited engagement with generated code, particularly among less experienced learners (Chow et al., 2026; Prather et al., 2024; Geng et al., 2026). In introductory programming, this creates a risk that students engage less with decomposition, testing, debugging, and semantic reasoning, allowing surface-level success to substitute for genuine understanding.

The broader literature presents a similarly mixed picture. Some studies warn that AI assistance may accelerate skill decay, hinder skill development, and reduce deep information processing through cognitive offloading (Macnamara et al., 2024; Kosmyna et al., 2025; Gerlich, 2025), while others suggest that students perceive generative AI as helpful for critical thinking and interpreting complex information (Alghamdi, 2025; Ravšelj et al., 2025). The key educational issue is therefore not whether generative AI is inherently beneficial or harmful, but under what conditions it supports learning without undermining deep thinking.

These tensions are particularly significant in programming education, where AI also complicates accountability and assessment integrity (Alanazi et al., 2025). If students can generate working code without being able to explain, test, or adapt it, then neither learning nor sound judgment has genuinely occurred. AI-assisted programming should therefore be treated as a pedagogical design problem rather than a simple adoption question. In response, this project proposes pedagogical guardrails: structured scaffolding before, during, and after AI use to support more deliberate, accountable, and educationally meaningful engagement

3. FRAMEWORK DEVELOPMENT

The framework presented in this paper was recently developed as a pedagogical response to a growing challenge in introductory programming: how to support the productive use of generative AI without leaving students unstructured or relying solely on tool-level restrictions. Kapoor (2026) shows that even when AI teaching assistants include guardrails designed to scaffold learning, many students still bypass them to access unconstrained outputs. This suggests that effective guardrails must be addressed through pedagogy, not only tool design.

The framework was informed by published preliminary observations from external colleagues, our own classroom observations of novice students using generative AI in programming tasks, and related work such as Geng et al. (2026), which identifies vague prompting, a focus on basic functionality, and limited engagement with generated code in novice AI-assisted programming. Across these sources, similar issues were visible: students were often prompted too early, engaged only superficially with generated code, and reflected little on what they had learned.

In response, the framework focuses on structuring how students engage with AI rather than restricting what AI can produce. The framework consists of five stages. **(1) Problem Decomposition** requires students to articulate the task structure and an initial solution approach before using AI. **(2) Programming Prompt Writing** asks students to express intent, requirements, and constraints clearly, treating prompting as a reasoning activity rather than a shortcut. **(3) Structured Prototype Interaction** encourages students to iteratively test and refine outputs rather than simply accepting them. **(4) Code Engagement** requires students to read, trace, debug, and interpret AI-generated code, countering black-box acceptance. **(5) Structured Retrospective** prompts students to review what worked, what failed, and how the solution evolved, helping to make AI interaction a source of learning rather than a hidden shortcut.

The framework is informed by cognitive load theory (Sweller, 1988), productive failure (Kapur, 2008), and the expertise reversal effect (Kalyuga et al., 2003). It provides a pedagogical model for designing AI-assisted programming activities that support more deliberate, accountable, and educationally meaningful use of generative AI in first-year computing.

4. WORKSHOP INTERVENTION AND INQUIRY DESIGN

The framework has been translated into a reusable workshop intervention embedded in two first-semester programming courses serving distinct first-year cohorts, one focused on theory and one more applied. This allows initial exploration across comparable but distinct introductory programming contexts.

The intervention is implemented through **Platypus**, a programming-learning platform initiated in early 2026 and still under active development by senior students as part of their capstone project at RMIT. This

provides an authentic teaching environment for integrating the workshop and enables rapid development of features needed for the intervention, including the capture of student prompts, reflections, interaction traces, and iterative artefacts.

The workshop includes an orientation to AI-assisted programming and its limitations, an introduction to the five-step framework, a hands-on programming task, and a structured reflective retrospective.

The inquiry is early-stage, exploratory, and practice-based, focusing on how the framework functions in classroom conditions, how students respond to the scaffolding, and how feasible the intervention is across first-year settings.

Evaluation will centre on an immediate post-workshop mixed-methods survey, combining confidence ratings across the five stages with open-ended responses. These data will be complemented by analysis of prompts, reflections, and interaction traces within Platypus. The pilot is intended to explore feasibility, usability, and early student response rather than large-scale effectiveness.

5. EMERGING INSIGHTS

As the project is still underway, results remain limited. However, several concerns have already emerged consistently through the literature review, framework development, classroom observation, and early design discussions.

Students often move directly to AI prompting without first decomposing the problem or articulating an initial solution approach. They may also confuse obtaining a plausible or working output with actually understanding the code, the solution process, or the limitations of the generated response (Chow & Peng, 2026). Validation and reflection likewise do not reliably occur unless they are deliberately designed into the learning activity. Without explicit scaffolding, students may engage only superficially with generated code, and accountability can remain hidden behind apparently successful outputs.

These early insights do not constitute final findings, but they clarify why the framework is needed and what the pilot is intended to examine more closely.

6. DISCUSSION

This paper argues that AI-assisted programming in first-year computing should be treated as a pedagogical design challenge rather than a simple question of permission or restriction. It proposes an early-stage framework and workshop model for deliberate, accountable AI-assisted programming in introductory contexts.

The project offers a middle path between uncritical adoption and blanket prohibition by positioning pedagogical guardrails as instructional scaffolding that supports purposeful AI use while preserving responsibility for problem analysis, code understanding, and judgment. It also highlights a changing role for educators in AI-rich environments: not only teaching programming content, but designing conditions for validation, reflection, and responsible AI use.

Although grounded in introductory programming, the work has broader relevance for AI literacy, assessment design, and computing education. As the pilot progresses, further refinement will be needed, but even at this early stage, the project offers a practical model for integrating generative AI into foundational computing education without sacrificing genuine student understanding.

7. REFERENCES

- Alanazi, M., Soh, B., Samra, H., & Li, A. (2025). The influence of artificial intelligence tools on learning outcomes in computer programming: A systematic review and meta-analysis. *Computers*, 14(5), 185. <https://doi.org/10.3390/computers14050185>
- Alghamdi, A. A. (2025). University students' perceptions of generative AI for critical thinking and creativity: The influence of self-efficacy and disciplinary differences. *Innovations in Education and Teaching International*. <https://doi.org/10.1080/14703297.2025.2600476>
- Chow, W., & Peng, S. (2026). "AI enhanced my critical thinking": Investigating the paradox of student perceptions and cognitive offloading in GenAI use. Melbourne University SoTL seminar.
- Geng, F., Shah, A., Li, H., Mulla, N., Swanson, S., Soosai Raj, G., Zingaro, D., & Porter, L. (2026). Exploring student-AI interactions in vibe coding. In *Proceedings of the 28th Australasian Computing Education Conference (ACE '26)* (pp. 45–54). Association for Computing Machinery. <https://doi.org/10.1145/3786228.3786236>
- Gerlich, M. (2025). AI tools in society: Impacts on cognitive offloading and the future of critical thinking. *Societies*, 15(1), 6. <https://doi.org/10.3390/soc15010006>
- Kalyuga, S., Ayres, P., Chandler, P., & Sweller, J. (2003). The expertise reversal effect. *Educational Psychologist*, 38(1), 23–31. https://doi.org/10.1207/S15326985EP3801_4
- Kapoor, A., Denny, P., ... Diaz, M. (2026). Exploring student behaviors and motivations when using AI teaching assistants with optional guardrails. In *Proceedings of the 28th Australasian Computing Education Conference (ACE '26)*. <https://doi.org/10.1145/3786228.3786233>
- Kapur, M. (2008). Productive failure. *Cognition and Instruction*, 26(3), 379–424. <https://doi.org/10.1080/07370000802212669>
- Kosmyna, N., Hauptmann, E., Yuan, Y. T., Situ, J., Liao, X.-H., Beresnitzky, A. V., Braunstein, I., & Maes, P. (2025). Your brain on ChatGPT: Accumulation of cognitive debt when using an AI assistant for essay writing task. *arXiv*. <https://doi.org/10.48550/arxiv.2506.08872>
- Macnamara, B. N., Berber, I., Çavuşoğlu, M. C., Krupinski, E. A., Nallapareddy, N., Nelson, N. E., Smith, P. J., Wilson-Delfosse, A. L., & Ray, S. (2024). Does using artificial intelligence assistance accelerate skill decay and hinder skill development.
- Ozkaya, I. (2023). The next frontier in software development: AI-augmented software development processes. *IEEE Software*, 40(4), 4–9. <https://doi.org/10.1109/MS.2023.3278056>
- Prather, J., Reeves, B. N., Leinonen, J., MacNeil, S., Randrianasolo, A. S., Becker, B. A., Kimmel, B., Wright, J., & Briggs, B. (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* (pp. 469–486). <https://doi.org/10.1145/3632620.3671116>
- Ravšelj, D., ... Aristovnik, A. (2025). Higher education students' perceptions of ChatGPT: A global study of early reactions. *PLOS ONE*, 20(2). <https://doi.org/10.1371/journal.pone.0315011>
- Sweller, J. (1988). Cognitive load during problem solving. *Cognitive Science*, 12(2), 257–285. https://doi.org/10.1207/s15516709cog1202_4