Coding with a Co-Pilot: A Systematic Review of Generative AI's Impact on Novice Programming Education

Shashank Reddy Srinivasa Reddy
Student
Minnesota State University Mankato
Mankato, MN 56001

Rushit Dave
Program Director
229 Wissink Hall
Mankato MN 56001

Mansi Bhavsar
Assistant Professor
222 Wissink Hall
Mankato MN 56001

ABSTRACT

The integration of Generative Artificial Intelligence (GenAI) tools into introductory programming (CS1) education challenges established pedagogical and assessment models. This systematic review synthesizes recent empirical literature, anchored by a meta-analysis of 32 controlled studies (2020-2024), to examine what GenAI assistance does and does not do for novice learning. The evidence reveals an efficiency-understanding paradox: relative to unassisted instruction, GenAI use significantly improves student performance scores (Standardized Mean Difference, SMD = 0.86), while gains in conceptual understanding are statistically negligible (SMD = 0.16, falling to -0.03 under sensitivity analysis). A five-profile taxonomy of novice-AI interaction indicates that learning impact depends on how students engage rather than on tool presence. This paper proposes an AI-integrated curriculum redesign framework grounded in constructive alignment, recalibrating learning outcomes, teaching activities, and assessment strategies toward process-based evaluation of the student's problem-solving journey.

Keywords

Generative AI; Introductory Programming (CS1); Cognitive Bypassing; Constructive Alignment; Process-Based Assessment; Computing Education

1. INTRODUCTION

Generative Artificial Intelligence (GenAI) tools such as ChatGPT and GitHub Copilot are now embedded in the daily practice of introductory computer science (CS1) students, and their association with elevated performance metrics is increasingly well documented [1-3]. The scale of this shift is best appreciated against its baseline: Mason et al.'s global survey of introductory programming courses framed by its own authors as the last snapshot of CS1 before code-generating tools became prevalent describes an instructional landscape that has, within two academic years, been overtaken [4]. In an environment of ubiquitous, low-cost access, prohibition is not a viable policy: longitudinal cohort data show tool familiarity reaching every student within a single semester (Section 4.2), and the same tools are already embedded in professional development practice [2].

This review therefore departs from binary performance measurement and interrogates the cognitive processes that GenAI assistance circumvents. Synthesizing controlled evidence, it documents that gains in what novices *produce* with GenAI are not matched by gains in what they *understand* an efficiency-understanding paradox with direct consequences for curriculum and assessment design. We use the term *cognitive bypassing* to describe the mechanism underlying this divergence: the routine, unnoticed circumvention of schema-forming work diagnostic debugging, incremental code articulation on which durable programming knowledge depends. Section 2 positions this term against adjacent constructs in the literature.

The paper makes three contributions. First, it consolidates current empirical evidence around the efficiency-understanding paradox, including sensitivity analyses that strengthen the finding. Second, it synthesizes behavioral studies into a taxonomy of novice-AI interaction profiles and the pedagogical risks each entails. Third, it translates both into an AI-integrated curriculum redesign framework grounded in constructive alignment [5, 6], recalibrating learning outcomes, teaching and learning activities, and assessment strategies toward evaluation of the student's problem-solving process rather than the final artifact alone.

2. RELATED WORK

2.1 The Shifting CS1 Landscape

Mason et al.'s survey of 91 introductory programming courses across 84 institutions in 18 countries provides the discipline's pre-GenAI baseline: a pedagogy organized around individually authored code and summative assessment of program correctness [4]. Becker et al., writing as code-generation models first reached usable quality, catalogued both the opportunities (on-demand exemplars, error explanation, reduced barriers to entry) and the challenges (academic integrity, over-reliance, licensing ambiguity) these tools pose, urging the community to adapt deliberately rather than reactively [1]. Denny et al.'s subsequent stock-taking established how far the ground had already shifted: models of the Codex generation solve most standard CS1 exercises, and on real CS1 exam questions such a model scored within the top quartile of the class, ranking 17th among 71 students [2]. Traditional artifact-based assessment is therefore structurally vulnerable, independent of any individual student's conduct. The same authors identify the field's open questions academic integrity, equitable access, and the reliability of generated code as matters requiring curricular rather than merely procedural answers [2].

2.2 Student-AI Interaction Patterns

How novices actually use these tools is better understood through a cluster of recent empirical studies. Prather et al.'s usability study of Copilot contributed two novel interaction patterns "drifting," in which novices move passively from one incorrect suggestion to the next, and "shepherding," in which they labor to steer auto-generated code toward a solution

instead of writing their own [7]. Ghimire and Edwards, logging keystrokes in two CS1 sections, observed that students most often query AI assistants for debugging help and conceptual explanation rather than complete solutions yet a large share of AI responses were copied directly into student code immediately after receipt [8]. Classroom-scale evidence shows the same duality at the cohort level: Zviel-Girshin's semester-long study of first-semester engineering students documents rapid, near-universal adoption and clear perceived benefits alongside concerns about cheating, over-reliance, and surface-level learning [9]. Güner and Er's three-session classroom study (N = 158) moved from behaviors to types, deriving five novice-AI interaction profiles from students' prompting patterns, ranging from fully AI-reliant code generation to complete AI independence [3]. This review adopts their taxonomy as an organizing device and extends it with a pedagogical risk synthesis (Section 4.3).

2.3 Positioning Cognitive Bypassing

Three established constructs border the phenomenon this review addresses. *Cognitive offloading* denotes the deliberate externalization of cognitive work onto tools or environments the learner knows what is being delegated. *Automation bias* denotes disproportionate trust in automated output at the moment of decision, accepting suggestions over one's own judgment. The *illusion of competence* describes the resulting self-assessment error; Prather et al. observed precisely this "unwarranted illusion of competence" among weaker students working with GenAI [10]. Cognitive bypassing, as used here, is distinct from all three: it names the upstream mechanism rather than the attitude or the self-assessment. Where offloading is deliberate, bypassing is involuntary and typically undetected by the student; where automation bias describes a trust error at a decision point, bypassing describes the silent removal of the schema-forming struggle debugging, tracing, incremental articulation from the learning process itself, with the illusion of competence [10] as its downstream consequence.

2.4 The Gap This Review Addresses

Practical guidance is accumulating: Mahon et al. distill emerging classroom practice into guidelines for GenAI's evolving role in introductory programming [11], and Denny et al. lay out a community agenda spanning integrity, equity, and tooling [2]. What this guidance lacks is a unifying instructional-design theory that connects revised objectives, activities, and assessment into one coherent structure. This review supplies that connection through constructive alignment [6], following Avouris et al.'s demonstration that Biggs's framework can organize a systematic CS1 course redesign for the GenAI era [5]. The contribution is thus not another set of recommendations but an aligned framework in which each recommendation has a defined structural role: outcomes specify what must be learned with and without assistance, activities move students toward the engagement profiles that support those outcomes, and assessments verify the outcomes in ways that generated artifacts cannot satisfy.

3. METHODOLOGY

3.1 Search Strategy and Review Design

This study is structured as an integrative literature review: a design that combines a statistical anchor an existing meta-analysis with primary empirical and theoretical studies evaluated qualitatively. The rationale is division of labor: statistical aggregation of controlled experiments already exists in a meta-analysis guided by the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) framework by Alanazi et al. [12], so this review's added value lies in qualitative thematic synthesis of the behavioral and pedagogical literature that effect sizes cannot capture, organized around the mechanisms behind the pooled results. Primary sources were identified across the ACM Digital Library, IEEE Xplore, ScienceDirect, and Google Scholar using Boolean combinations of terms including "generative AI," "ChatGPT," "Copilot," "CS1," "novice programmer," and "introductory programming," restricted to peer-reviewed, English-language publications from 2020 to 2024.

3.2 PICO Framework

Source evaluation followed the PICO structure. The **P**opulation comprises novice programmers in CS1 or equivalent introductory courses. The **I**ntervention is the use of GenAI tools, principally ChatGPT and GitHub Copilot. The **C**omparator is manual coding without AI assistance all effect sizes reported in Section 4 are relative to unassisted instruction. The **O**utcomes are student performance scores, task-completion time, and conceptual understanding or retention.

3.3 Inclusion and Exclusion Criteria

Studies were included if they reported empirical observation of novice programmers using GenAI tools, presented a validated pedagogical framework applicable to CS1, or synthesized such evidence. Excluded were studies of classical (pre-generative) AI tutoring systems, studies of advanced or professional populations, and opinion pieces without empirical or theoretical grounding. Candidate sources were screened on title and abstract against the PICO criteria, with full-text review determining final inclusion. The resulting corpus comprises the anchoring meta-analysis [12], ten empirical or synthesis studies of GenAI in novice programming [1-4, 7-11, 13], and two instructional-design sources [5, 6].

3.4 Quality Assessment

Study quality for the quantitative evidence base was evaluated using the criteria applied by the primary meta-analysis [12]: the Newcastle–Ottawa Scale for non-randomised studies, the JBI Checklist for experimental designs, and the RoB 2 tool for risk of bias. Alanazi et al. report that randomised controlled trials and quasi-experimental designs were the most common approaches among their 32 included studies [12]. For the primary studies selected directly by this review, relevance and rigor were appraised against the PICO criteria above; heterogeneity statistics (I^2) are reported alongside every pooled effect in Section 4 so that readers can weigh the stability of each estimate.

4. RESULTS

4.1 Learning Efficiency vs. Conceptual Understanding

The synthesized evidence reveals a consistent divergence termed here the efficiency-understanding paradox between what GenAI-assisted novices produce and what they retain. Across the 32 controlled studies aggregated by Alanazi et al., GenAI assistance significantly improved student performance scores relative to unassisted instruction (SMD = 0.86, 95% CI [0.36, 1.37], $p = 0.0008$, $I^2 = 54\%$) [12]. Task-completion time showed a moderate pooled reduction (SMD = -0.69) that was not statistically significant ($p = 0.34$) and carried extreme heterogeneity ($I^2 = 95\%$); the interval estimate is omitted here because the source's published interval is internally inconsistent (Section 5.4). The efficiency gains of GenAI assistance are therefore established for graded performance, not for speed.

The understanding side of the paradox is starker. Pooled gains in success and ease of understanding were statistically negligible (SMD = 0.16, 95% CI [-0.23, 0.55], p = 0.41, I^2 = 55%), and the estimate was unstable: Alanazi et al.'s sensitivity analysis shows that removing a single study shifts it to -0.03 (p = 0.83) effectively zero [12]. Improved artifacts, in other words, coexist with unimproved comprehension, creating a risk of masking conceptual gaps beneath syntactically correct code. Figure 1 presents all four pooled outcomes on a common scale.

Three features of Figure 1 warrant emphasis. First, the confidence interval for performance scores excludes zero across its full range, whereas the intervals for both conceptual-understanding estimates straddle it: the divergence between the two outcomes is not a difference of magnitude but the difference between an effect reliably present and one indistinguishable from no effect. Second, heterogeneity varies sharply by outcome. The two effects with interpretable intervals show substantial but comparable heterogeneity (I^2 = 54% and 55%), whereas task-completion time reaches I^2 = 95%, indicating that the constituent studies disagree almost completely; this is the basis for the present review's refusal to advance any claim about speed. Third, the sensitivity result is decisive for interpretation a pooled estimate that moves from 0.16 to -0.03 upon the removal of a single study cannot support a claim of modest benefit, however qualified. Alanazi et al. additionally assessed publication bias across outcomes using funnel plots and Egger's test [12].

The perception data sharpen the contrast further. The pooled prevalence of students reporting the tools useful and beneficial was 1.00 (95% CI [0.92, 1.00]) with zero heterogeneity (I^2 = 0%) a complete consensus, and the most statistically stable finding in the evidence base [12]. Near-unanimous satisfaction thus coexists with conceptual gains that do not survive sensitivity analysis. Student experience, in this literature, measures the tools' appeal rather than their learning value, and satisfaction data should not be read as evidence of pedagogical effectiveness. Behavioral evidence corroborates the mechanism: many AI responses are pasted directly into student programs immediately after receipt [8], and Prather et al. found that while well-prepared, high-self-efficacy students used GenAI to accelerate toward solutions, weaker students struggled while maintaining an unwarranted illusion of competence—evidence that these tools may widen the gap between well and poorly performing students [10].

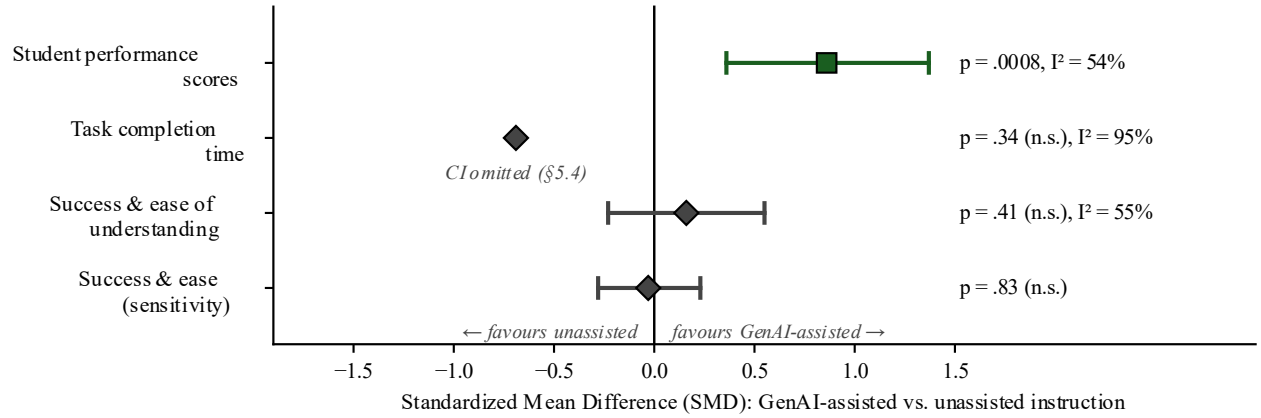

Figure 1. Pooled effects of GenAI-assisted versus unassisted CS1 instruction across 32 controlled studies, with 95% confidence intervals, p-values, and heterogeneity (I^2); data from Alanazi et al. [12]. Squares denote statistically significant effects, diamonds non-significant ones. The bottom row shows the leave-one-out sensitivity estimate for conceptual understanding. The task-completion interval is omitted because the published interval is inconsistent with its point estimate (Section 5.4).

Table 1. Meta-analytic outcomes of GenAI-assisted CS1 learning relative to unassisted instruction; data from Alanazi et al. [12].

Outcome	Effect size (SMD)	95% CI	p-value	Heterogeneity (I^2)
Student performance scores	0.86	[0.36, 1.37]	0.0008	54%
Task completion time	-0.69	*	0.34 (n.s.)	95%
Success & ease of understanding	0.16	[−0.23, 0.55]	0.41 (n.s.)	55%
Perceived usefulness & benefits**	1.00	[0.92, 1.00]	-	0%

* Interval omitted: the interval as published is internally inconsistent with its point estimate and p-value (see Section 5.4).

** Pooled prevalence (proportion of students reporting the tools useful), not a standardized mean difference; no p-value is reported in the source.

4.2 Demographics and Generalizability

Adoption data indicate that these dynamics are not confined to early enthusiasts. In a 12-week mixed-methods study of 73 teams of engineering students, Zviel-Girshin observed AI tool familiarity climb from 28% to 100% of students within the single semester [9]. The same cohort reported high satisfaction with the tools and broad use for routine tasks, mirroring the meta-analytic perception findings above [9]. Universal exposure, however, is the only safely generalizable finding: the cohort represents one institution and one discipline, and outcome patterns may differ across contexts a caution reinforced by the high heterogeneity of the pooled effects in Section 4.1. What the adoption curve does establish is that any pedagogical design premised on students *not* using these tools is already obsolete.

4.3 Taxonomy of Novice-AI Interaction Profiles

Aggregate effects conceal wide variation in how individual novices engage AI assistance. Güner and Er's analysis of student prompting behavior identified five interaction profiles, reproduced in Table 2 with this review's synthesis of the dependency level and pedagogical risk each implies [3]. The profiles matter pedagogically because they are malleable rather than fixed traits: in the same cohort, the share of students operating as AI-Collaborative Coders rose from 11.0% in the first session to 51.4% by the third, following explicit training in prompting skills [3].

Figure 2 shows the full redistribution behind that headline figure. The movement is systematic rather than confined to a single profile: AI-Reliant Code Generators fell from 30.8% to 11.6% across the three sessions, and AI-Reliant Generator & Refiners peaked at 31.5% in the second session before falling to 21.9% in the third, while the collaborative profile absorbed the majority of the migration. McNemar-Bowker tests confirm that the redistribution between consecutive sessions is statistically significant (Session 1 to 2: $\chi^2(10, N = 135) = 34.478$, $p < .001$; Session 2 to 3: $\chi^2(10, N = 136) = 57.099$, $p < .001$) [3]. The shift is therefore a measured intervention effect rather than incidental variation.

Learning impact, in short, is determined by the mode of engagement rather than the presence of the tool and the mode of engagement is teachable. The reliant end of the spectrum, where output is accepted without review, is cognitive bypassing in operation; the collaborative middle preserves the student's agency over core logic while capturing productivity benefits. The taxonomy also refines the equity concern raised by Prather et al. [10]: if weaker students disproportionately settle into reliant profiles while stronger students collaborate, aggregate performance gains will conceal a widening distribution of durable skills making profile-aware instruction, not tool policy alone, the operative lever.

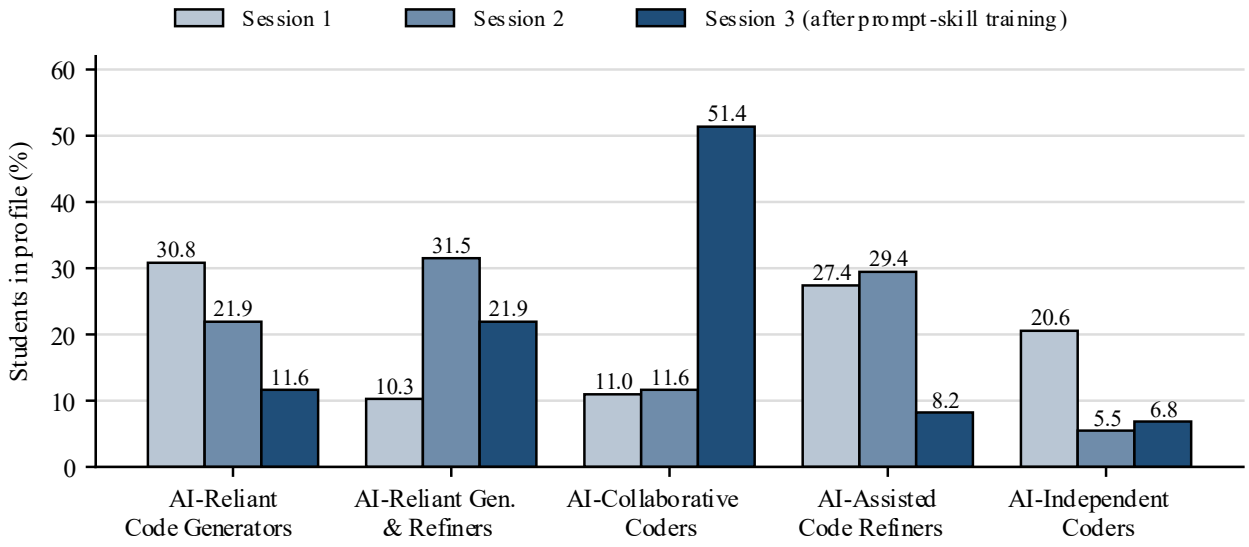

Figure 2. Distribution of students across the five interaction profiles in each of three classroom sessions (N = 146 per session); data from Güner & Er [3]. Explicit prompt-skill training preceded the third session.

Table 2. Novice-AI interaction profiles identified by Güner & Er [3] (N = 158, three-session classroom study). Behavioral characteristics paraphrase the source; dependency levels and pedagogical implications are this review's synthesis.

Interaction profile	Dependency level*	Defining behavioral characteristics	Pedagogical implication*
AI-Reliant Code Generators	Severe	Passive prompting; task instructions pasted for direct answers; minimal engagement with the problem-solving process	High risk of skill decay and illusion of competence [10]; requires intervention
AI-Reliant Code Generator & Refiners	High	Code generated via AI; AI used recursively to fix errors; no manual debugging	Impedes formation of mental models of control flow and program state
AI-Collaborative Coders	Moderate	Iterative partnership; student retains agency over core logic; AI used for scaffolding	Balances productivity and learning; mirrors professional workflows
AI-Assisted Code Refiners	Low	Manual authoring first; AI queried for specific fixes or optimizations	Supports schema formation; AI functions as targeted supplemental tutor
AI-Independent Coders	None	Complete avoidance due to distrust, integrity concerns, or learning goals	Develops traditional skills but may lack AI-collaboration literacy

* Columns marked with an asterisk are this review's synthesis, not classifications made by Güner & Er [3].

5. DISCUSSION: AI-INTEGRATED CURRICULUM REDESIGN FRAMEWORK

The results indicate that GenAI's risks to novice learning are conditional concentrated in reliant engagement modes and that engagement modes respond to instruction. A structural response is therefore feasible. This section proposes one, anchored in constructive alignment [6]: the principle that learning outcomes, teaching and learning activities (TLAs), and assessment must be designed as a mutually reinforcing system, recently demonstrated as an organizing frame for GenAI-era CS1 redesign by Avouris et al. [5].

5.1 Redefining Learning Outcomes

Emerging guidance emphasizes actively adapting learning objectives to the GenAI environment, understanding tool-usability paradigms, and exploring controlled AI-generated exercises to scaffold novice understanding [7, 11, 13]. Within a constructively aligned course, this recalibration yields three outcome families:

- **Independent proficiency.** Students demonstrate fundamental programming concepts tracing, writing, and debugging code entirely without AI assistance.

- **AI-augmented proficiency.** Students critically inspect, debug, and modify AI-generated code, which requires strategic problem decomposition to direct the tool effectively [2].
- **Ethical awareness.** Students explain code bias, algorithmic fairness, and the professional and societal implications of deploying AI-generated artifacts.

5.2 Modifying Teaching and Learning Activities

TLAs should follow a progressive structure that deliberately moves students along the taxonomy of Section 4.3 away from reliant profiles and toward collaborative ones. Early modules pair foundational syntax with AI-ethics discussion while assistance is constrained, protecting initial schema formation. As students progress to complex data structures, instructors incorporate live demonstrations that model effective prompting and critical evaluation of responses. Capstone activities center on collaborative critique: teams evaluate AI-generated solutions for correctness, style, and societal impact. The 51.4% collaborative-profile share achieved after prompt-skill training [3] indicates that such deliberate instruction, not mere exposure, drives the shift.

5.3 Adapting Assessment Strategies

Assessment is the framework's critical joint, because traditional summative assessment grading only final code correctness is structurally subverted by models that score within the top quartile of real CS1 exam takers [2]. Constructive alignment requires assessing the process by which students arrive at solutions. Three integrity-related adjustments come first:

- **Reclassifying misconduct.** Because large language models lack human agency, submitting unreviewed generated code is structurally the use of an unauthorized resource rather than plagiarism of a person; policies should say so explicitly.
- **Retiring detection-based enforcement.** Conventional plagiarism detectors are ineffective against non-deterministic generated output; enforcement premised on detection is no longer credible.
- **Teaching licensing obligations.** Students may unknowingly incorporate generated code that carries open-source license-compliance obligations, a risk they must be taught to recognize [1].

Beyond integrity, three assessment redesigns operationalize process-based evaluation. AI-integrated rubrics grade whether tool use followed course policy, was cited, and was accompanied by rigorous critique of the generated code's logic. Prompt-engineering exercises reposition articulation of the problem not production of the artifact as the graded skill, steering students toward the collaborative profiles of Section 4.3 [3]. Hybrid feedback models let AI tools serve as preliminary syntax checkers while instructors evaluate the student's metacognitive reflection through brief structured code walkthroughs, revision histories, or written rationales directly countering the illusion of competence documented among AI-assisted novices [10]. Each assessment mode verifies one outcome family from Section 5.1: unassisted checkpoints verify independent proficiency, critique-based rubrics verify AI-augmented proficiency, and reflection components verify ethical awareness.

5.4 Limitations and Future Research

Four limitations bound these conclusions. First, the anchoring meta-analysis exhibits high statistical heterogeneity ($I^2 = 95\%$ for task-completion time) and internal reporting inconsistencies including the interval estimate omitted in Section 4.1 so its pooled effects warrant conservative interpretation [12]. Second, one primary source could be verified at abstract level only, as its full text was inaccessible during this review [8]. Third, scalable process-based assessment remains logistically demanding: oral walkthroughs are resource-prohibitive for large CS1 cohorts, motivating research into secure AI-driven agents for automated process evaluation. Fourth, there is a notable gap in longitudinal evidence [10]: current literature over-indexes on immediate task performance, leaving open whether bypassed learning in CS1 manifests as skill decay in subsequent courses, and no validated curriculum yet exists for systematically teaching prompt engineering to novices.

6. CONCLUSION

The integration of GenAI into introductory programming education is a permanent shift in the discipline's operating conditions, not a transient trend. The synthesized evidence shows that large language models deliver real efficiency benefits for novices while posing a significant pedagogical challenge: absent structural adaptation, their routine use bypasses the productive struggle on which deep conceptual learning depends, leaving fragile mental models beneath improved deliverables.

The response this review proposes is neither prohibition nor unrestricted access but realignment. Anchored in constructive alignment, the AI-integrated framework recalibrates learning outcomes to distinguish independent from AI-augmented proficiency, sequences teaching activities to move students toward collaborative engagement profiles, and shifts assessment from artifact correctness to the student's documented reasoning process. The taxonomy evidence is encouraging on this point: engagement modes are teachable, and deliberate prompt-skill instruction measurably moves novices toward profiles that preserve intellectual agency. Students entering an AI-augmented software industry are not served by courses that ignore these tools, and they are not served by courses that let the tools do the learning's work; the framework's purpose is to hold both truths at once.

Future research should prioritize longitudinal designs that track whether AI-assisted novices retain computational thinking into subsequent coursework, the development of validated prompt-engineering curricula, and review methodologies that distinguish generative tools from the classical AI systems with which methodologically limited syntheses still conflate them. The framework offered here gives computing educators a systematic path for that adaptation one in which each policy, activity, and rubric has a defined role in protecting how novices learn to reason about programs.

7. REFERENCES

[1] Becker, B. A., Denny, P., Finnie-Ansley, J., Luxton-Reilly, A., Prather, J., and Santos, E. A. 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 (SIGCSE TS 2023)*, 500–506. DOI: 10.1145/3545945.3569759.

[2] Denny, P., Prather, J., Becker, B. A., Finnie-Ansley, J., Hellas, A., Leinonen, J., Luxton-Reilly, A., Reeves, B. N., Santos, E. A., and Sarsa, S. 2024. Computing education in

the era of generative AI. *Communications of the ACM* 67, 2, 56–67. DOI: 10.1145/3624720.

[3] Güner, H. and Er, E. 2025. AI in the classroom: Exploring students' interaction with ChatGPT in programming learning. *Education and Information Technologies* 30, 12681–12707. DOI: 10.1007/s10639-025-13337-7.

[4] Mason, R., Simon, Becker, B. A., Crick, T., and Davenport, J. H. 2024. A global survey of introductory programming courses. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education (SIGCSE TS 2024)*. DOI: 10.1145/3626252.3630761.

[5] Avouris, N., Sgarbas, K., Caridakis, G., and Sintoris, C. 2025. Teaching introduction to programming in the times of AI: A case study of a course re-design. arXiv preprint arXiv:2508.06572. DOI: 10.48550/arXiv.2508.06572.

[6] Biggs, J. 1996. Enhancing teaching through constructive alignment. *Higher Education* 32, 3, 347–364.

[7] Prather, J., Reeves, B. N., Denny, P., Becker, B. A., Leinonen, J., Luxton-Reilly, A., Powell, G., Finnie-Ansley, J., and Santos, E. A. 2023. "It's weird that it knows what I want": Usability and interactions with Copilot for novice programmers. *ACM Transactions on Computer-Human Interaction* 31, 1, Article 4. DOI: 10.1145/3617367.

[8] Ghimire, A. and Edwards, J. 2024. Coding with AI: How are tools like ChatGPT being used by students in foundational programming courses. In *Artificial Intelligence in Education (AIED 2024)*, Lecture Notes in Computer Science, vol. 14830. Springer, Cham, 259–267. DOI: 10.1007/978-3-031-64299-9_20.

[9] Zviel-Girshin, R. 2024. The good and bad of AI tools in novice programming education. *Education Sciences* 14, 10, 1089. DOI: 10.3390/educsci14101089.

[10] Prather, J., Reeves, B. N., Leinonen, J., MacNeil, S., Randrianasolo, A. S., Becker, B. A., Kimmel, B., Wright, J., and 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 (ICER 2024)*. DOI: 10.1145/3632620.3671116.

[11] Mahon, J., Mac Namee, B., and Becker, B. A. 2024. Guidelines for the evolving role of generative AI in introductory programming based on emerging practice. In *Proceedings of the 2024 Conference on Innovation and Technology in Computer Science Education (ITiCSE 2024)*. DOI: 10.1145/3649217.3653602.

[12] Alanazi, M., Soh, B., Samra, H., and 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. DOI: 10.3390/computers14050185.

[13] Speth, S., Meißner, N., and Becker, S. 2023. Investigating the use of AI-generated exercises for beginner and intermediate programming courses: A ChatGPT case study. In *Proceedings of the 35th IEEE Conference on Software Engineering Education and Training (CSEE&T 2023)*. DOI: 10.1109/CSEET58097.2023.00030.