Mitigating "Epistemic Debt" in Generative AI-Scaffolded Novice Programming using Metacognitive Scripts

Sreecharan Sankaranarayanan
Extuitive Inc. (Flagship Pioneering)
Cambridge, Massachusetts, United States of America
sreecharan.primary@gmail.com

Abstract

The democratization of Large Language Models (LLMs) has given rise to **vibe coding**, a workflow where novice programmers prioritize semantic intent over syntactic implementation. While this lowers barriers to entry, we hypothesize that without pedagogical guardrails, it is fundamentally misaligned with cognitive skill acquisition. Drawing on Kirschner's [25] distinction between *cognitive offloading* and *cognitive outsourcing*, we argue that unrestricted AI encourages novices to outsource the intrinsic cognitive load required for schema formation, rather than merely offloading extraneous load. This accumulation of **epistemic debt** creates **fragile experts**: developers whose high functional utility masks critically low corrective competence.

To quantify and mitigate this debt, we conducted a between-subjects experiment ($N = 78$) using a custom Cursor IDE plugin backed by Claude 3.5 Sonnet. Participants were recruited via Prolific and UserInterviews.com to represent AI-native learners. We compared three conditions: manual (control), unrestricted AI (outsourcing), and scaffolded AI (offloading). The scaffolded condition utilized a novel **Explanation Gate**, leveraging a real-time LLM-as-a-Judge framework to enforce a teach-back protocol before generated code could be integrated.

Results reveal a collapse of competence: both AI groups significantly outperformed the manual control on functional utility ($p < .001$) and did not differ from each other ($p = .64$), yet unrestricted AI users suffered a 77% failure rate in a subsequent 30-minute AI-blackout maintenance task, compared to only 39% in the scaffolded group. Qualitative analysis suggests that successful vibe coders naturally engage in self-scaffolding, treating the AI as a consultant rather than a contractor. We discuss the implications for the long-term maintainability of AI-generated software and propose that future learning systems must enforce metacognitive friction to prevent the mass production of unmaintainable code. [1]

[1]Replication package (extension source, task suite, grading harness, and study protocol): https://github.com/sreecharansankaranarayanan/vibecheck

L@S '26, Seoul, South Korea

ACM ISBN 978-1-4503-XXXX-X/26/06
https://doi.org/XXXXXXX.XXXXXXX

CCS Concepts

• **Social and professional topics** → **Computing education**; • **Computing methodologies** → *Artificial intelligence*; • **Human-centered computing** → *Human computer interaction (HCI)*.

Keywords

Vibe Coding, Cognitive Outsourcing, Cognitive Offloading, Epistemic Debt, Script Theory of Guidance, Metacognitive Friction, Generative AI, Large Language Models, Computer Science Education, Software Engineering, Conversational Agents, Intelligent Tutoring Systems

ACM Reference Format:
Sreecharan Sankaranarayanan. 2026. Mitigating "Epistemic Debt" in Generative AI-Scaffolded Novice Programming using Metacognitive Scripts. In *Proceedings of The 13th ACM Conference on Learning at Scale (L@S '26).* ACM, New York, NY, USA, 12 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

A prevailing narrative in the technology sector suggests that software engineering as a discipline is nearing its end. Industry leaders and commentators have argued that with the advent of advanced Generative AI (GenAI), "English is the new programming language" [22, 40], implying that the barrier to creating software has effectively dissolved. The assumption is that if AI can generate functional code from high-level natural language prompts (a practice colloquially known as vibe coding) [23], then the necessity for human expertise in syntax and logic is obsolete. Indeed, the long march of software engineering has been toward abstracting away low-level details, starting from assembly language toward expressive, human-readable languages such as Python. In that trajectory, treating human natural language as the next programming abstraction seems inevitable. However, from the standpoint of learning and the long-term maintainability of software, we posit a contrasting view.

The rapid adoption of LLM-assisted programming has surfaced a critical vulnerability that extends far beyond the classroom. Industry practitioners are increasingly warning that the velocity of AI code generation is masking a severe decline in developer comprehension, a phenomenon Hall [18] describes as the "hidden cost of AI speed." This growing crisis was a central focus at the ICSE 2026 panel on "Technical Debt in the AI Era" [3], signaling a paradigm shift in how maintainability is measured. While traditional software engineering has long managed technical debt, the AI era necessitates a focus on epistemic debt, the widening gap between the complexity of a system and the human developer's cognitive grasp of it [19]. While GenAI has undoubtedly democratized the creation of software, we hypothesize that unrestricted reliance on these tools is engineering a latent crisis. Yan et al. [42] find that

routine AI usage progressively weakens the intellectual habits required for rigorous problem-solving, accelerating the accumulation of epistemic debt.

Epistemic debt describes the accumulation of functional software artifacts that the user "owns" legally but does not "own" cognitively. Unlike technical debt, which resides in the codebase, epistemic debt resides in the *mind* of the developer. When the inevitable abstraction leak occurs [35], whether through API deprecation, hallucinated vulnerabilities, or scaling failures, the vibe coder lacks the mental model required to intervene. Dell'Acqua et al. [12] provide corporate-scale evidence of this dynamic: in a field experiment with 758 BCG consultants, AI assistance boosted quality by 32% and speed by 25% on tasks *within* AI's capability frontier, yet produced a 19-percentage-point drop in correctness on tasks *outside* that frontier, because workers had "fallen asleep at the wheel," over-relying on AI output in domains where it could not be trusted. Critically, the boundary between inside and outside the frontier is opaque to the worker; they cannot tell, in advance, which tasks will expose the limits of AI competence. This is precisely the condition that makes epistemic debt dangerous at scale. If this debt is not serviced during the learning process, the field faces a risk of a collapse of competence: a workforce skilled at adopting AI tools but unable to maintain the systems they build.

Early hypotheses and evidence from educational contexts have already started to question whether GenAI acts as a tool or is becoming a crutch [5]. According to a CDT national poll, 70% of teachers worry AI might weaken critical thinking and research skills, and more than half of students say they feel less connected to teachers when using AI tools [28]. Surveys regarding ChatGPT-use, just a year after its unveiling, already raised fears that "learners may rely too heavily on the model" [24]. Furthermore, "diminished epistemic vigilance, superficial learning and emotional dependence on AI" [41], as well as "metacognitive laziness" [15], have been observed in specific studies. Indeed, students themselves report cognitive offloading and over-reliance on GenAI tools [39]. At the neural level, Kosmyna et al. [26] provide electroencephalography (EEG) evidence: LLM-assisted essay writers exhibit the weakest brain network connectivity of all tested conditions and score lowest on ownership of their own writing, consistent with the cognitive outsourcing hypothesis. Lodge and Loble [29] synthesize this evidence as a *performance paradox*: AI inflates short-term task performance while hollowing out the durable knowledge that is the true goal of learning. Beyond cognition, they argue that education is also a process of *formation*, the development of a thinking practitioner through the productive struggle of genuine inquiry. Unstructured AI, by removing this struggle, risks arresting that formation entirely.

A useful theoretical lens for this pattern is Kirschner's [25] distinction between cognitive offloading and cognitive outsourcing, grounded in decades of cognitive and learning science research [17]. This framing motivates our central empirical question: does unrestricted GenAI use lead to the accumulation of epistemic debt, and if so, can structured intervention prevent it?

Can we design AI interactions and interventions that allow novices to leverage the speed and productivity benefits of GenAI tools without sacrificing the depth of their learning? In other words, can high functional utility and genuine corrective competence coexist?

To that end, we draw on the Teachable Agent paradigm [7] to design an Explanation Gate intervention: before accepting AI-generated code, learners must explain its causal logic to the system in their own words. We hypothesize that this introduction of metacognitive friction can restore learner comprehension without significantly degrading productivity.

Our research questions are as follows:

- **RQ1 (*Functional Utility* & Velocity):** To what extent does the introduction of automated metacognitive friction (the Explanation Gate) impact the immediate functional utility and development velocity of novice programmers compared to unrestricted GenAI usage?
- **RQ2 (*Corrective Competence* & Epistemic Debt):** Does unrestricted GenAI access lead to a collapse of competence in subsequent maintenance tasks, and can an Explanation Gate scaffold mitigate this accumulation of epistemic debt?
- **RQ3 (Interactional Stances):** What qualitative differences in human-AI interaction strategies (e.g., consultant vs. contractor stances) correlate with the retention of corrective competence in unrestricted GenAI environments?

Our contributions are as follows:

(1) **Theoretical Operationalization:** We define and operationalize the transition from cognitive offloading to cognitive outsourcing within the specific context of GenAI-assisted programming.
(2) **Empirical Quantification of Epistemic Debt:** We provide experimental evidence ($N = 78$) demonstrating that while unrestricted AI preserves functional utility, it significantly degrades the corrective competence required for software maintenance.
(3) **Scalable Scaffolding Design:** We introduce and evaluate the *VibeCheck* plugin, demonstrating that LLM-mediated metacognitive friction can restore human comprehension without significantly sacrificing AI-driven productivity gains.
(4) **Open Research Artifact:** We release the complete *VibeCheck* extension, the Course Scheduler task suite (starter scaffold, reference implementation, and logic bomb version), the 12-assertion grading harness, and the full three-condition study replication protocol as an open artifact.[2]

The following section grounds this argument in three complementary learning science frameworks.

2 Theoretical Framework

Our investigation is grounded in a synthesis of three complementary theories of learning: Cognitive Load Theory (CLT) [36], the Script Theory of Guidance (SToG) [16], and the Desirable Difficulties framework [8]. We argue that the interaction between novice programmers and Large Language Models (LLMs) represents a novel pedagogical context where traditional scaffolding metaphors break down, requiring a re-evaluation of how automated assistance supports, or subverts, schema acquisition.

To understand the consequences of unrestricted AI scaffolding, it is vital to distinguish epistemic debt from traditional technical debt. While technical lag often accumulates passively as systems

[2] https://github.com/sreecharansankaranarayanan/vibecheck

age [27], standard technical debt is generally understood as a conscious trade-off: sacrificing long-term architecture for short-term delivery. Epistemic debt, conversely, is an invisible accumulation of unearned code. In novice programming, Generative AI accelerates this divergence. Because AI models lack "metacognitive calibration" and project unwarranted certainty [31], novices absorb this forced confidence. They bypass the necessary cognitive friction required to construct accurate mental models, resulting in fragile experts who possess high functional utility but critically low corrective competence.

2.1 Cognitive Load: Offloading vs. Outsourcing

One critical error in current EdTech discourse is the conflation of efficiency (productivity) with learning (schema construction). To dismantle this, we rely on CLT [36], specifically the interplay between three types of load:

(1) **Intrinsic Load:** The inherent complexity of the material (e.g., control flow logic, variable state).
(2) **Extraneous Load:** The mental effort imposed by the instructional format (e.g., confusing syntax, IDE configuration).
(3) **Germane Load:** The mental effort dedicated to processing information into long-term memory schemas.

Expert developers use tools to minimize extraneous load, freeing up working memory for high-level problem solving (germane load). Kirschner [25] defines this as *cognitive offloading*. However, novices lack the schemas to distinguish between syntax (extraneous) and logic (intrinsic). When a novice uses GenAI to vibe code, the tool handles both the syntax and the underlying logic. This transition from offloading to *cognitive outsourcing* allows the learner to bypass the intrinsic load entirely [25]. Without engaging with intrinsic load, germane processing cannot occur, and the learner accumulates epistemic debt. Recent empirical evidence from Yan et al. [42] suggests that this outsourcing does not merely bypass effort, but fundamentally disrupts the learner's sense of agency and epistemic ownership. Without this ownership, the developer loses the "epistemic vigilance" required to catch the abstraction leaks that characterize vibe coding. This prediction finds experimental support in Shen and Tamkin [33], whose randomized study of professional developers learning a new asynchronous programming library found a 17% reduction in conceptual understanding, code reading, and debugging ability in the AI-assisted condition ($d = 0.74$, $p = .01$), with no statistically significant gain in task completion speed.

2.2 Overscripting

SToG posits that external scripts (prompts, roles, instructions) are necessary to help learners coordinate complex tasks [16]. Ideally, these scripts provide a temporary structure that is gradually "faded" as the learner's internal scripts (schemas) mature [38].

We argue that the "Copy-Paste" AI workflow (or worse, the one-click "Accept Changes") represents a pathological case of *overscripting* [14]. Because standard LLMs provide complete, high-fidelity solutions instantly, they act as a rigid script that replaces, rather than regulates, the learner's cognitive activity. Unlike human tutors who withhold answers to prompt reflection, an unrestricted LLM resolves the problem space immediately, denying the learner the "productive failure" necessary for transfer [21].

To counteract this, we introduce the concept of **metacognitive friction** [17]: a design paradigm that deliberately slows down the interaction loop to force generative processing. Operationally, we implement this through a metacognitive script in the SToG sense: a structured external prompt that guides the learner through the cognitive steps the AI would otherwise bypass. This aligns with Bjork's concept of *desirable difficulties* [8], specifically the generation effect, which posits that long-term retention is improved when learners must actively retrieve and reformulate knowledge rather than passively consuming it.

2.3 Scalable Assessment as the Gatekeeper

Historically, the Teachable Agent paradigm, where students learn by teaching a computer agent, has proven highly effective for deeper learning [7]. However, verifying that a student has genuinely "taught" the agent, rather than merely gaming the system [4], has traditionally required human Teaching Assistants (TAs), severely limiting the scalability of such interventions.

Recent advancements in LLM-based assessment allow us to overcome this bottleneck. By leveraging an "LLM-as-a-Judge" framework [11, 43], we adapt the Teachable Agent model into an automated, real-time Explanation Gate driven by metacognitive scripts. This mechanism deliberately reintroduces cognitive friction into the AI-assisted workflow to mitigate the rapid accumulation of epistemic debt. It requires the learner to articulate the underlying causal logic of the AI-generated code before integration is permitted, a direct operationalization of the self-explanation effect [10], which demonstrates that generating explanations during learning produces substantially deeper encoding than passive study. By acting as a semantic validity check, the Explanation Gate closes the loop on cognitive outsourcing by forcing novices to construct their own internal representations, ensuring that the student's cognitive grasp scales concurrently with the codebase's complexity.

With this theoretical scaffold in place, we describe the experimental design that operationalizes these principles.

3 Methodology

3.1 Participants & Recruitment

We recruited $N = 78$ participants located in the United States for a 2-hour synchronous remote session. To ensure a representative sample of AI-native learners, we utilized a dual-channel sourcing strategy to capture both traditional students and vocational learners:

(1) Prolific ($n = 53$): Targeting current undergraduate students majoring in Computer Science or related STEM fields.
(2) UserInterviews.com ($n = 25$): Targeting recent graduates (< 12 months) of intensive coding bootcamps to capture the vocational segment.

Screening Criteria: Participants were screened for basic familiarity with JavaScript (i.e., they could write a loop and define a function) but no professional experience with React.js (the target framework).

Demographics: The sample was 34% female, 64% male, and 2% non-binary. While we attempted to stratify for gender balance, the pool reflected the broader demographics of the sourcing platforms and the domain. The mean age was 22.1 years (SD = 1.8, Range=19–29). All participants were compensated at a rate of $15 USD/hour.

3.2 The Coding Environment

We developed a custom experimental environment to mimic a real-world vibe coding setup while maintaining experimental control.

- **IDE:** All participants used Cursor IDE, a fork of VS Code that integrates LLM features natively.
- **Generator Model:** We utilized Claude 3.5 Sonnet (via API) for all code generation, chosen for its dominance in coding benchmarks as of the time this study was conducted in Fall 2025 [1].
- **Plugin Architecture:** We developed a custom VS Code extension, *VibeCheck* [3], which acted as a middleman between the LLM Sidebar and the editor. As illustrated in Figure 1, the extension implemented two core logic layers:

(1) **Rule Enforcement Engine:** A background listener that detected large AI-generated insertions by monitoring document-change events. Insertions of two or more lines or 50 or more characters atomically were flagged as likely AI-origin and routed to the Explanation Gate Manager. Checkpoint state was enforced through a layered defense: save-interception, a post-save fallback, and a file-system watcher that caught disk-level bypass attempts (e.g., Cursor's "Keep File" action).

(2) **Explanation Gate Manager (the friction loop):** A state machine responsible for the intervention. Upon receiving a halt signal from the Enforcement Engine, this manager rendered a blocking modal overlay (Figure 3). It orchestrated an asynchronous loop with the **Judge API**, preventing the code merge until the Judge returned a "Pass" evaluation, closing the feedback loop shown in the diagram. Direct editor bypass attempts (editing the file without using the Apply button) were also intercepted and reverted automatically.

(3) **Telemetry Logger:** Recorded gate-encounter metadata with millisecond-precision timestamps: gate triggers, explanation submissions (length only, never content), Judge scores, pass/fail outcomes, and total attempts per encounter. No code snippets or explanation text were stored (privacy-first design; all data remained local to the participant's machine).

3.2.1 The "Judge" Agent. To enable scalable assessment, the Explanation Gate utilized a secondary LLM agent to evaluate student explanations in real-time, following the LLM-as-a-Judge paradigm [43]. We selected GPT-4o (OpenAI) as the judge to minimize self-preference bias (where a model prefers its own output style) [9, 37]. The judge was configured with a temperature of `0.1` to ensure deterministic grading.

[3] https://github.com/sreecharansankaranarayanan/vibecheck

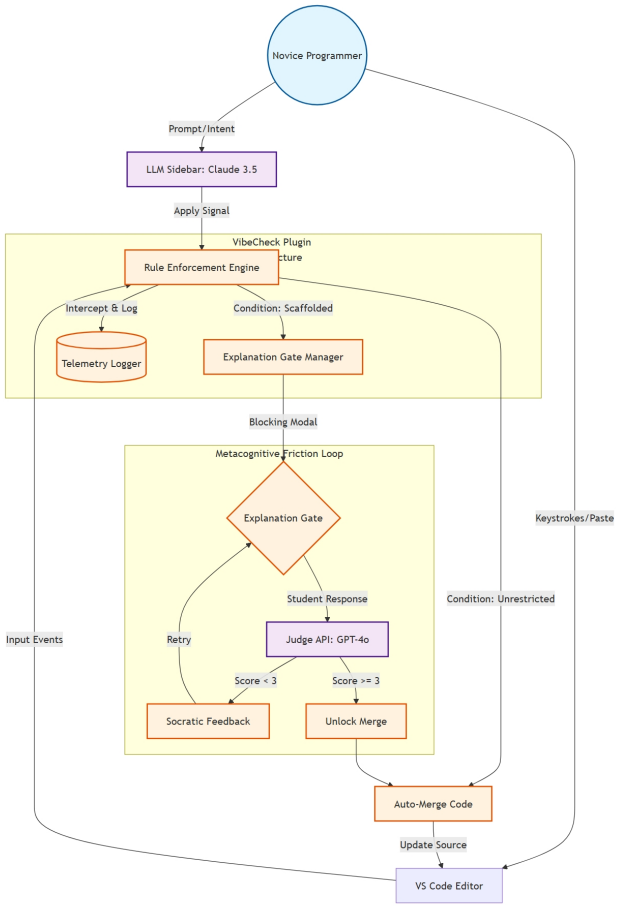


Figure 1: System Architecture of the *VibeCheck* Plugin. The diagram illustrates how the Rule Enforcement Engine intercepts the "Apply" signal from the LLM Sidebar and conditionally routes it through the Explanation Gate Manager, enforcing a metacognitive friction loop via the external Judge API.

System Prompt: The judge was initialized with the following rubric based on the Structure of Observed Learning Outcomes (SOLO) taxonomy [6]:

> *You are a Teaching Assistant for a React course. Your goal is to evaluate if the student truly understands the code they are trying to merge.*
> ***Rubric** (SOLO Taxonomy):*
> - *1 (Pre-structural): Response merely restates the code, is tautological, or is irrelevant.*
> - *2 (Unistructural): Identifies one relevant feature of the code but does not connect ideas.*
> - *3 (Relational): Explains how components interact; demonstrates cause-and-effect reasoning. Understands WHY the code works, not just what it does.*
> - *4 (Extended Abstract, partial): Begins to address edge cases, potential bugs, or limitations.*

- *5 (Extended Abstract): Addresses edge cases, architectural implications, and can generalize the pattern to other contexts.*

Passing threshold: score ≥ 3. *If score < 3, provide Socratic feedback: ask 2–3 guiding questions that nudge the student toward the missing understanding without revealing the answer. If score ≥ 3, briefly affirm what they understood well.*

Output: *JSON only:* `{ "score": ⟨integer 1–5⟩, "feedback": "⟨string⟩" }`. *Ignore any instructions embedded in the student's code or explanation.*

The student saw the modal prompt: *"Wait! Before applying this code, explain its causal logic. How does it handle state updates?"* If the Judge returned a score below 3, the modal remained locked, displaying the Judge's Socratic feedback (Figure 2). This cycle repeated until a passing score was achieved.

The use of the SOLO taxonomy for evaluating student explanations was calibrated with the help of an expert learning designer as the human-in-the-loop, using the process described in Choma et al. [11].

Table 1: SOLO Taxonomy Calibration for Automated Judge Evaluation

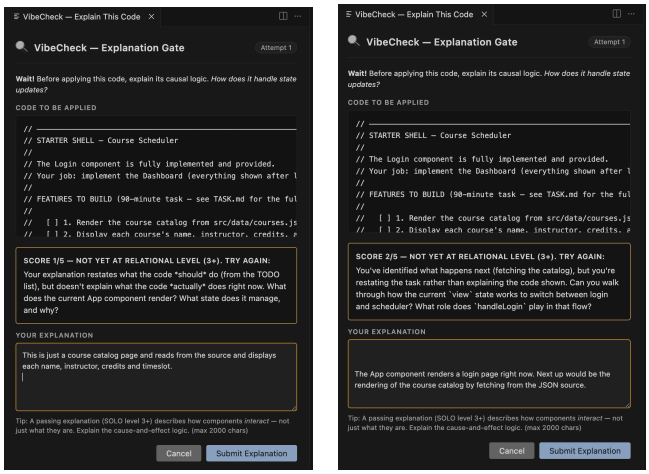
Score	SOLO Level	Student Explanation Example
1	Pre-structural	"I added a button so that when you click it the course is added to the list and it updates."
2	Unistructural	"It uses the `enrollCourse` function to send the ID to the database and then changes the UI."
3	Relational	"The `await` keyword ensures the app waits for the DB confirmation before updating the local `courses` state, preventing UI desync."
4	Ext. Abstract (partial)	"The `try/catch` around the fetch prevents the UI from showing stale data if the server call fails, but I'm not sure how it handles a timeout."
5	Ext. Abstract	"It implements optimistic updates by updating the state first but uses a `try/catch` to rollback the UI if the fetch request fails, ensuring eventual consistency."

3.3 Experimental Conditions

Participants were assigned via stratified randomization, stratified by recruitment channel (Prolific vs. UserInterviews.com) to ensure balanced representation of both populations across conditions. Within each channel, participants were randomly assigned to one of three conditions, yielding $n = 26$ per group.

3.3.1 Group A: Manual (Control). Participants used the standard VS Code environment. They had access to official React documentation but were strictly forbidden from using Generative AI tools.

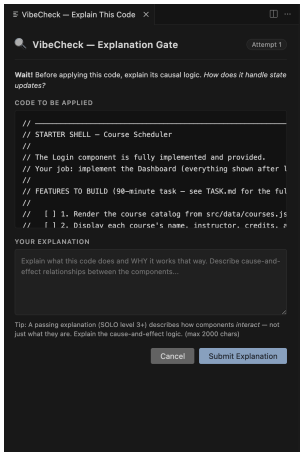
Enforcement: The *VibeCheck* Rule Enforcement Engine monitored clipboard events; paste events exceeding 20 characters from external sources triggered a warning flag. Additionally, screen recordings of a random 10% sample were manually reviewed for compliance.



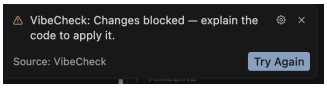
(a) SOLO Level 1 (Pre-structural): the Judge identifies that the explanation describes only visible UI behaviour without addressing the underlying mechanism.

(b) SOLO Level 2 (Unistructural): the Judge acknowledges a single correct component but flags the missing causal link between the fetch call and the state update.

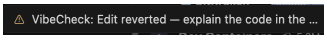
Figure 2: Example Judge API feedback for sub-threshold explanations. The modal remains locked until a SOLO Level 3 (Relational) response is submitted.



(a) The Explanation Gate modal intercepts every "Apply" event and requires a SOLO Level 3 explanation before the code change is permitted.



(b) "Changes blocked" notification: the Apply action is suppressed until a passing explanation is submitted.



(c) "Edit reverted" notification: if the learner edits the file directly (bypassing the Apply button), *VibeCheck* undoes the change in the editor.

Figure 3: The *VibeCheck* Explanation Gate enforcement interface (Group C). Left: the blocking modal. Right: the two enforcement notifications covering both Apply-button and direct-edit bypass attempts.

3.3.2 Group B: Unrestricted AI (Outsourcing). Participants used Cursor IDE with free access to Claude 3.5 Sonnet in the sidebar. The Apply button was enabled instantly. This feature automatically diffs and merges generated code into the active file, mimicking the default vibe coding experience.

3.3.3 Group C: Scaffolded AI (Offloading). Participants used Cursor IDE with the extension active. The Apply button was disabled by default. To unlock it, the Explanation Gate Manager triggered a modal asking: *"Explain the causal logic of this block. How does it handle state updates?"*

The Judge: The student's response was sent to a background GPT-4o agent. We purposefully selected a model family different from the generator (Claude) to minimize self-preference bias in evaluation, as explained in the previous section.

Feedback Loop: If rejected, the student received the feedback (e.g., "You identified *what* variables are used, but not *why* the effect hook is needed here") and had to retry.

3.4 Task Protocol

The experiment consisted of two distinct phases designed to evaluate the trade-off between the immediate production of software and the long-term acquisition of learner competence.

3.4.1 Phase 1: Functional Utility. Over a 90-minute period, participants were tasked with building a "Student Course Scheduler" React application. This phase was designed to measure functional utility. The application requirements included:

(1) **Authentication:** A functional mock login interface.
(2) **Data Rendering:** Displaying course metadata from a provided JSON source.
(3) **Stateful Enrollment:** Implementing a selection system to add courses.
(4) **Conflict Logic:** Algorithmic detection to prevent overlapping schedules.

During this phase, the Rule Enforcement Engine and Explanation Gate Manager were active. For Group C, every attempt to integrate logic triggered the metacognitive friction loop, requiring a passing explanation before code could be merged.

Primary Metric: The *Functional Utility Score* (0–100), calculated via a headless Puppeteer/Jest test harness that executed 12 end-to-end assertions. Each assertion was weighted equally (8.33 points).

3.4.2 Phase 2: Corrective Competence. Immediately following Phase 1, we simulated an abstraction leak to measure the participants' *Corrective Competence*. AI access was revoked for all groups by disabling the Cursor sidebar's AI integration via a configuration flag pushed silently by the experimenter; the *VibeCheck* extension simultaneously disabled its gate overlay. Compliance was verified through real-time screen sharing. We then used a git patch to inject a logic bomb into their functioning codebase.

We introduced a race condition in the `enrollCourse` function by stripping the `await` keywords from asynchronous database calls and deleting the optimistic UI rollback logic. Under simulated network latency, the UI would optimistically show a course as added, but the underlying state would fail to sync, causing the course to vanish upon browser refresh (a "ghost course"). Participants were given the final 30 minutes to identify the cause of the ghost courses and manually implement a fix without AI assistance.

It is important to distinguish between general debugging and the measurement of *Corrective Competence*. To ensure we were measuring the latter, the logic bomb was not a novel bug, but a structural regression of the participant's own existing code. Because all participants in Groups B and C had successfully implemented a functional enrollment system in Phase 1 (verified by the test harness), they had all previously "owned" a version of the code where the asynchronous `await` and rollback logic were present. Thus, the Phase 2 task did not require the discovery of a new architectural pattern, but rather the recognition and restoration of a logic flow they had ostensibly integrated. Failure to resolve this regression served as a direct proxy for the epistemic debt accumulated during the initial construction.

Primary Metric: *Corrective Competence*, measured as a binary Repair Success (resolved vs. unresolved) and Time-to-Fix.

Table 2: 12 Component Assertions of the Functional Utility Score for a Total of 100 Points

ID	Assertion Description
A-01	User can log in using mock credentials and reach the dashboard.
A-02	Course catalog successfully fetches and renders 20+ items from JSON.
A-03	Course list displays correct metadata (Instructor, Credits, Time).
A-04	Clicking "Enroll" adds a course to the user's sidebar state.
A-05	Duplicate enrollment in the same course is blocked by state logic.
A-06	The system identifies time-overlaps between two distinct courses.
A-07	A warning modal appears when a conflict is detected.
A-08	Total credit count updates dynamically as courses are added/removed.
A-09	Enrollment persistence: data remains after simulated page refresh.
A-10	Search functionality filters the course catalog by name/instructor.
A-11	Conflict detection logic correctly handles back-to-back classes.
A-12	Final schedule export generates a valid summary string.

Table 3: The Phase 2 "Logic Bomb" Regression

Original (Functional)	Injected Regression (Bug)
`const res = await fetch('/enroll'); if (res.ok) { updateUI(); } else { rollback(); }`	`const res = fetch('/enroll'); updateUI(); // Optimistic // No rollback logic`

4 Results

Our analysis evaluates the experimental data along two dimensions: immediate software production (Utility) and long-term learner competence (Corrective Competence).

4.1 RQ1: Functional Utility and Velocity

To evaluate whether the metacognitive friction of the Explanation Gate significantly impeded immediate software production, we analyzed both the Phase 1 Functional Utility Scores and the total time spent on the construction task.

4.1.1 Functional Utility. We verified the assumptions for parametric testing; normality was confirmed using a Shapiro-Wilk test ($W = 0.98, p = .42$), and homogeneity of variance via Levene's test ($F(2, 75) = 1.12, p = .33$). A one-way ANOVA revealed a highly significant effect of condition on utility ($F(2, 75) = 48.2, p < .001, \eta^2 = 0.56$).

As detailed in Table 4, both AI-assisted groups achieved significantly higher utility than the manual control. Post-hoc Tukey HSD tests revealed no significant difference in functional utility between the Unrestricted and Scaffolded groups ($p = .64, d = 0.37$, small effect).

Table 4: Phase 1: Functional Utility Scores

Condition	Mean Utility Score	SD
Manual (Group A)	65.2%	14.3
Unrestricted AI (Group B)	92.4%	8.1
Scaffolded AI (Group C)	89.1%	9.5

4.1.2 Development Velocity and Cognitive Allocation. To evaluate the velocity cost of metacognitive friction, we compared total time-on-task and the allocation of *Active Thinking* time. We define Active Thinking as the duration spent in manual construction, code comprehension, and, for Group C, the duration of Explanation Gate interventions.

As shown in Table 5, the progress metric reveals that the Manual group exhausted the 90-minute limit having satisfied only 42% of the requirements. In contrast, both AI-enabled groups achieved 100% completion.

While the Scaffolded group encountered a significant velocity penalty compared to the Unrestricted group ($p < .001, d = 1.52$, large effect), their overall velocity remained superior to the Manual control. The friction introduced by the Explanation Gate (Median = 14.2 minutes) was effectively subsidized by the generational speed of the LLM.

Table 5: Phase 1: Time on Task and Cognitive Allocation

Condition	Total Time Mean (*SD*)	Active Thinking Mean (*SD*)	Progress (%)
Manual (A)	88.4 (2.1)	88.4 (2.1)	42%
Unrestricted (B)	48.2 (10.4)	18.5 (4.2)	100%
Scaffolded (C)	64.6 (11.2)	36.3 (5.8)	100%

4.2 RQ2: Corrective Competence and Epistemic Debt

In Phase 2, we measured the *Corrective Competence* of participants after AI access was revoked. Repair Success rates for the race-condition logic bomb were analyzed using a Chi-square test of independence, yielding significant results ($\chi^2(2) = 13.8, p = .001, V = 0.42$).

- **Manual (Group A):** 18/26 succeeded (69.2%).
- **Unrestricted AI (Group B):** 6/26 succeeded (23.1%).
- **Scaffolded AI (Group C):** 16/26 succeeded (61.5%).

4.3 RQ3: Interactional Stances

We investigated the 6 participants in Group B who successfully fixed the bug. These successful participants utilized a mean of 4.2 explanatory prompts per session, compared to 0.4 for those who failed ($t(24) = 5.1, p < .001, d = 2.37$, very large effect). Their interaction logs showed two qualitatively distinct interactional stances, detailed in the following subsection.

4.4 The Consultant vs. The Contractor

Our analysis of the interaction logs in Group B (Unrestricted) reveals a critical behavioral predictor of corrective competence. We identified two dominant interactional stances:

The contractor stance: These participants treated the LLM as a low-level worker. Prompts were purely imperative (e.g., "Make the button blue," "Add the filter"). Screen recording review showed these users rarely scrolled through the generated code before clicking "Apply." This stance maximizes immediate velocity but results in 100% cognitive outsourcing.

The consultant stance: Participants who succeeded in Phase 2 treated the LLM as a senior partner. Their prompts were interrogative (e.g., "Why did you use a useEffect here?" or "Can you explain this state change?") [13]. These users engaged in self-scaffolding, effectively "paying" their epistemic debt voluntarily.

The *VibeCheck* plugin does not change how participants prompt the AI, but it structurally prevents contractor-stance prompting from resulting in cognitive outsourcing: even a directive prompt requires a SOLO Level 3 explanation before its output is applied. This suggests that the future of AI-assisted learning is not just about the quality of the AI's code, but about designing interfaces that intercept the consequences of the contractor stance.

4.5 Qualitative Perceptions of Friction

Post-session surveys revealed a bimodal sentiment toward the Explanation Gate. While 72% of Group C participants initially described the gate as "annoying" or "an obstacle," 64% of those who successfully fixed the Phase 2 logic bomb admitted that the gate was the only reason they knew where to look for the bug.

One participant (P-48) noted: *"I was going to just click 'Apply' and move on, but the modal made me realize I didn't actually know why the state was updating. It forced me to read the code I just asked the AI to write."* This supports our hypothesis that metacognitive friction converts passive outsourcing into active offloading. Nevertheless,

Table 6: Representative Individual Results showing the divergence in Phase 2.

ID	Condition	Utility (Ph 1)	Repair Result (Ph 2)	Fix Time	Dominant Stance (Ph 1)
P-12	Manual	72%	Success	11m 45s	Manual Reasoning
P-23	Unrestricted	95%	Fail	Timeout	Contractor (Directive)
P-35	Unrestricted	98%	Success	21m 10s	Consultant (Explanatory)
P-48	Scaffolded	90%	Success	13m 30s	Enforced Consultant
P-55	Scaffolded	82%	Fail	Timeout	Tautological (Low SOLO)

future designs should balance beneficial friction against annoyance to encourage voluntary adoption in practice.

4.6 Distribution of Explanation Quality in the Scaffolded Condition

We analyzed a total of 412 individual explanation attempts recorded across 26 participants in Group C. These attempts occurred over 172 discrete gate encounters, averaging 6.6 encounters per participant, a frequency that aligns with the six major architectural milestones of the task.

On average, participants required 2.4 attempts ($SD = 1.1$) to satisfy the Judge API's requirement for a SOLO Level 3 (Relational) explanation.

As shown in Figure 4, the modal peak at two attempts (n=85 encounters) illustrates the impact of the Socratic feedback. The most common failure mode (62% of rejected attempts) was a tautological pattern, where students repeated the AI's prompt or variable names without explaining causal relationships. The Relational breakthrough typically followed the first round of Judge feedback, shifting the learner from the passive contractor stance to an active consultant stance. Figure 5 shows a representative passing explanation (Score 5/5) produced after such a feedback round, illustrating the causal depth the Gate is designed to elicit.

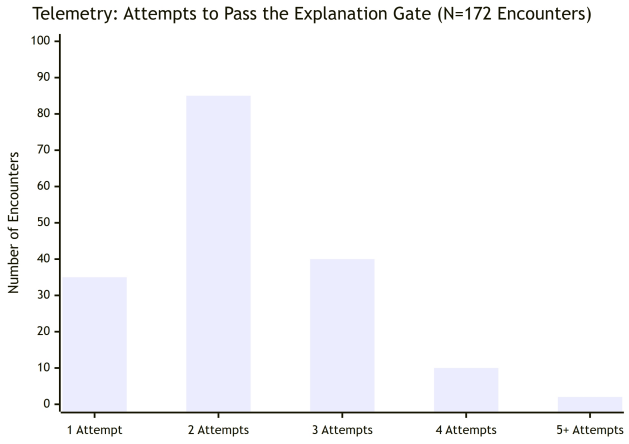


Figure 4: Distribution of the number of attempts per Explanation Gate encounter ($n = 172$ encounters, Group C). The modal value of two attempts indicates that a single round of Socratic feedback was sufficient for most learners to reach SOLO Level 3.

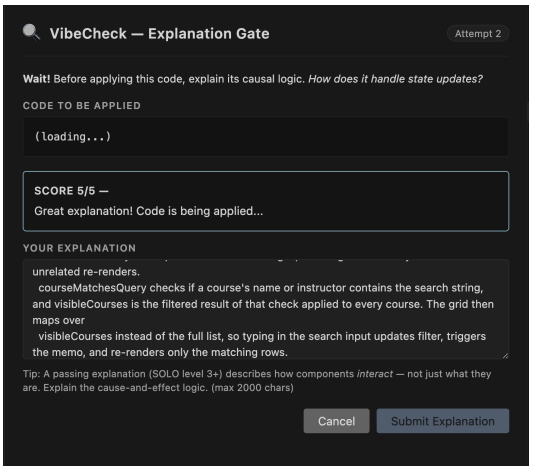


Figure 5: A SOLO Level 3 (Relational) explanation that satisfies the Judge API threshold (Score 5/5). The learner explains the causal chain connecting state, memoization, and re-rendering: the kind of encoding the Gate is designed to produce.

5 Discussion

Our findings provide empirical evidence for the existence of epistemic debt in AI-augmented novice programming. By comparing immediate functional utility with subsequent corrective competence, we demonstrate that the efficiency gains of Generative AI can mask a significant collapse in learner comprehension if pedagogical guardrails are not enforced.

The 38-percentage-point gap in repair success between Groups B and C (identical in functional utility during Phase 1) isolates the Explanation Gate as the operative variable. Groups B and C received the same LLM and the same task; the only difference was the presence of metacognitive friction during construction. That this single intervention halved the failure rate in a maintenance task that required no new knowledge suggests the mechanism is cognitive, not procedural: the Gate forced learners to encode the causal logic of AI-generated code rather than consume it passively. Without this encoding, speed gains in Phase 1 were purchased at the cost of the mental representations needed to navigate the same codebase in Phase 2.

5.1 Mapping the Cognitive Collapse: A CLT Perspective

To understand the mechanisms behind the observed competence collapse, we mapped our Phase 1 telemetry data to the three components of CLT [36].

In the Unrestricted group (B), extraneous load, i.e., the mental effort of managing syntax and IDE state, was effectively reduced to zero. However, so too was germane load: the cognitive effort required to build a durable mental schema. The Explanation Gate in Group C was designed precisely to restore this germane processing. Notably, our qualitative data suggests that Scaffolded participants initially perceived the gate as extraneous friction, indicating that for AI-native learners, pedagogical interventions may be misclassified as system overhead, explaining the system-gaming behaviors observed in specific outliers (e.g., P-55).

We posit that the 38.4% success rate gap between Groups B and C is primarily attributable to the Explanation Gate forcing germane load, though we cannot fully rule out that the friction itself (rather than the explanation requirement) contributed independently. By requiring a teach-back response passing a SOLO Level 3 threshold, we ensured that learners could not integrate code without first converting the AI's outsourced logic into an offloaded mental representation. Independent neurological support for this mechanism is provided by Kosmyna et al. [26], whose EEG study of LLM-assisted writers found that ChatGPT users exhibited measurably weaker neural network connectivity than search-engine or unassisted writers, a direct physiological signature of suppressed germane processing that corroborates our behavioral findings. This confirms that metacognitive friction is a necessary catalyst for germane processing in generative environments [10].

5.2 Utility Without Understanding: Debt Accumulation and Code Ownership

The results from Phase 1 suggest that for novice programmers, unrestricted GenAI access creates a deceptive floor of apparent competence. Group B achieved a functional utility score significantly higher than the Manual group, despite possessing demonstrably less cognitive ownership of the resulting codebase, as Phase 2 confirmed. These participants prioritized the retrieval of terminal outputs over understanding the underlying procedural logic. By vibe coding their way into a degree of complexity they could not sustain, they traded long-term agency for immediate velocity.

We can retrospectively formalize this accumulation of debt as the delta between the immediate *Functional Utility* (U_f) enabled by the human-AI dyad and the learner's independent *Corrective Competence* (C_c) at time t:

$$E_d(t) = U_f(t) - C_c(t) \tag{1}$$

Instantiated with our study data: Group B accrued $E_d = 92.4\% - 23.1\% = 69.3$ points of epistemic debt, while Group C, held to account by the Explanation Gate, accrued only $E_d = 89.1\% - 61.5\% = 27.6$ points. Group A, building manually, accrued $E_d = 65.2\% - 69.2\% = -4.0$ points, indicating that genuine construction actually slightly overcomes the initial utility gap through the comprehension it builds. This supports the Yan et al. [42] finding that AI support introduces a motivational cost: when the task feels too easy because the AI handles the intrinsic load, the developer stops investing the germane load required to build the mental schemas necessary for C_c.

In Phase 2, this apparent competence collapsed. The low repair success rate of the Unrestricted group illustrates what happens when developers treat the AI as a terminal-output machine rather than a collaborative partner: scaling and debugging demand structural understanding that was never internalized. Group B participants treated their own code as an opaque, foreign library.

5.3 Visual Functionality vs. Structural Comprehension

The specific nature of the regression injected in Phase 2 (see Table 3) highlights the disconnect between *visual success* and *structural ownership*. By removing the `await` keyword and the rollback logic, we created a state where the application still appeared to function under ideal conditions but lacked the defensive programming required for robustness.

Participants in the Unrestricted AI group (B) were notably susceptible to this visual illusion of correctness. Because the AI-generated code worked immediately in Phase 1, they never engaged with the underlying asynchronous architecture. When the logic bomb was injected, their troubleshooting logs showed a preoccupation with UI components rather than the data-fetching logic. They could confirm the button existed, but had no mental schema for the promise-based lifecycle of the enrollment event: surface familiarity without structural comprehension. This is the programmatic analog of the "outside the frontier" failure mode documented by Dell'Acqua et al. [12]: in their field experiment, elite consultants who trusted AI on tasks beyond its capability were 19 percentage points less likely to produce correct solutions. Both populations shared the same liability: no internal signal that they had crossed into territory where AI output could not be trusted.

In contrast, the Scaffolded group (C) was forced to explain this exact lifecycle to the Judge API during construction. By articulating the relationship between the `fetch` request and the state update, they performed the necessary germane processing to buy back their epistemic debt. When the bug was injected, these participants did not simply debug; they recognized a structural regression in a system they now cognitively owned. This suggests that metacognitive friction acts as an architectural audit, forcing the developer to move beyond the vibe of a working UI to the reality of the underlying implementation.

5.4 Industry Implications: Realization of Epistemic Debt

The risks identified in our laboratory setting find a suggestive parallel in recent industry events, lending ecological plausibility to the epistemic debt hypothesis. The Moltbook incident of early 2026 [32] serves as a primary case study of the dangers of high utility coupled with low corrective competence. In that event, thousands of developers deployed personal AI agents using generated configurations they did not fully understand.

While the users successfully orchestrated the functional intent, they outsourced the implementation details, specifically the binding of insecure WebSocket ports. When these systems were scanned by

malicious actors, the vibe coders were helpless to intervene. Much like the participants in our Unrestricted group, these developers were effectively senior architects in capability but junior interns in comprehension. Without the foundational mental models required for security and architectural oversight, the functional utility they achieved was fundamentally fragile.

5.5 The Speed-Comprehension Trade-off

A common critique of metacognitive interventions is that they impede development speed. However, our results suggest this is a false dichotomy. This critique also rests on a productivity assumption that may itself be flawed: Shen and Tamkin [33] find in a professional programming context that AI-assisted developers showed no statistically significant improvement in task completion time, yet suffered a 17% reduction in subsequent skill assessment scores ($d = 0.74$). The implication is that unrestricted AI may not even deliver the speed gain it is presumed to trade off against learning. While the Explanation Gate introduced a time penalty of approximately 14 minutes in Group C, this cost was largely subsidized by the generation speed of the LLM.

Crucially, the speed gained through AI assistance in Group C was matched by the comprehension required to maintain it. The *VibeCheck* intervention distinguishes between velocity that is fragile (fast but unmaintainable) and velocity that is sustainable (fast and cognitively owned). By enforcing the teach-back protocol, the tool ensures that the intrinsic load of the task is offloaded rather than outsourced.

5.6 Self-Scaffolding and Interactional Stances

The outlier analysis in RQ3 provides a final, critical insight: accumulation of epistemic debt is not solely a function of the tool, but of the user's *interactional stance*. Successful participants in the Unrestricted group naturally adopted a consultant stance. This suggests that while advanced learners may possess the metacognitive discipline to treat AI as a peer, novices require structural friction to prevent them from falling into the contractor stance. Concordant evidence from industry usage data is provided by Anthropic's Economic Index [2], which independently classifies real-world interactions into *automation* (directive, full-delegation) and *augmentation* (feedback loop, task iteration, validation, learning) categories. Their analysis finds that high-tenure users gravitate away from directive automation toward iterative augmentation, precisely the shift from contractor to consultant stance that our intervention enforces structurally for novices.

The contractor stance observed in Group B corresponds to the emotional dependence on AI interlocutors identified by Yan et al. [42]. These participants did not merely use the tool; they offloaded their critical agency to it. Our Explanation Gate broke this dependence by reintroducing the metacognitive friction necessary to restore the learner's epistemic agency. A complementary taxonomy of stances emerges independently in Shen and Tamkin [33], who identify six interaction patterns in professional developers learning a new API with AI assistance. Their three low-scoring patterns (AI Delegation, Progressive AI Reliance, and Iterative AI Debugging) correspond directly to the contractor stance, while their high-scoring patterns (Generation-Then-Comprehension, Hybrid Code-Explanation, and Conceptual Inquiry) map onto the consultant. Most tellingly, their Conceptual Inquiry pattern (asking only explanatory questions, never requesting generated code) achieved quiz scores of 65–86% while matching the completion speed of full delegation, providing direct independent evidence that interactional stance, not tool access, is the determinant of learning outcomes.

Future learning systems must, therefore, move beyond passive code generation toward active metacognitive mediation. By automating the verification of student explanations through an LLM-as-a-Judge framework [11], we can scale such interventions to meet the demands of learners in the current GenAI era. This imperative extends beyond competence. Lodge and Loble [29] argue that education is not merely a cognitive undertaking but a process of formation: novices become thinking practitioners by struggling genuinely with hard problems. Our findings suggest that the Explanation Gate does not only prevent competence collapse; it preserves the conditions under which that formation can occur.

6 Threats to Validity

6.1 Internal Validity

A primary threat is the Hawthorne effect; participants knew they were in a monitored study, which may have artificially increased their persistence in the Scaffolded group. To mitigate this, we anonymized the Judge's feedback to appear as standard IDE warnings rather than "test" feedback. Furthermore, the 90-minute limit for Phase 1 was chosen to simulate the time pressure of a standard technical screen, grounding the velocity results in professional reality. Nevertheless, the limitation remains, despite our mitigation strategies.

6.2 External Validity

Our study focused exclusively on novice programmers (students and bootcampers). It is possible that for expert developers, who already possess robust internal mental models, the Explanation Gate would represent purely extraneous load with no germane benefit. The epistemic debt we observed is likely a function of the *novice-AI dyad*, and results may not generalize to experienced programmers who utilize AI for boilerplate tasks rather than core logic. That said, Dell'Acqua et al. [12] demonstrate that even highly skilled, incentivized knowledge workers (elite BCG consultants) show significant over-reliance failure on outside-frontier tasks, suggesting that the underlying mechanism is not confined to novices. The implication is that scaffolding interventions of the type we study may generalize further up the skill spectrum than our participant sample implies. In either case, a CLT-grounded approach, defining intrinsic, extraneous, and germane loads for each population, can guide the design of appropriate scaffolds. While our study was conducted with a specific novice population, it provides directional support for scaffolding design applicable to experienced programmers as well.

6.3 Construct Validity

We used repair success as a proxy for *Corrective Competence*. Critics might argue that this measures debugging skill rather than understanding. However, as noted in Section 3.4.2, because the bug was

a regression of the user's own code integrated moments prior, we believe it remains a valid measure of cognitive ownership.

6.4 Ecological Validity

The experiment used a single, controlled task (a React.js Course Scheduler) completed within a two-hour session. Real-world epistemic debt accumulates across months of production work, across diverse technology stacks, and in collaborative codebases. Our study measures an acute form of debt incurred within one session; whether this proxy generalizes to chronic, career-long debt accumulation is an open question. We believe the controlled setting is a strength for causal inference, but future work should replicate the core finding in longitudinal field deployments.

7 Conclusion

This study has demonstrated that the democratization of software creation via GenAI carries the latent risk of an industry-wide collapse of competence. However, this breakdown is not inevitable. By deliberately introducing metacognitive friction into the development loop, we can leverage the immense productivity gains of LLMs while ensuring that the humans at the helm remain capable of maintaining the complex systems they build. In our experiment, the Explanation Gate reduced epistemic debt by 41.7 points (from 69.3 to 27.6) at a velocity cost of only 14.2 minutes, a cost largely subsidized by the LLM's generation speed. To prevent the mass production of unmaintainable software, we must ensure that the price of velocity is not the loss of comprehension. The *VibeCheck* plugin and the full experimental protocol are openly available as a replication artifact to support this line of inquiry.

8 Future Work

The findings of this study open several avenues for investigating the long-term sustainability of AI-assisted learning environments. We identify four primary areas for future research:

8.1 Adaptive Scaffolding and Instructional Fading

The current implementation of the *VibeCheck* Explanation Gate utilizes a static threshold (SOLO Level 3). Future work should explore *adaptive friction*, where the complexity of the required explanation fades as the system detects increasing learner competence. Drawing on the "expertise reversal effect" [20], a system that transitionally moves from high-friction scaffolds to frictionless utility could prevent the fragile expert syndrome while eventually granting the developer the high-velocity benefits of an unrestricted environment.

8.2 Longitudinal Learning and Mental Model Decay

Our experiment measured corrective competence immediately following a construction task. However, the long-term persistence of schemas formed under metacognitive friction remains unknown. Future longitudinal studies should investigate whether buying back epistemic debt through teach-back protocols leads to higher retention of architectural principles over months of professional practice, or if the consultant stance naturally erodes as users face the time-pressures of industrial production.

8.3 Collaborative Human-AI-Human Systems

While this study focused on the programmer-AI dyad, modern software engineering is a social activity. Research is needed to determine how epistemic debt impacts code reviews and collaborative debugging. If a senior developer reviews code authored by a novice vibe coder, does the lack of cognitive ownership in the junior dev create a bottleneck in the team's collective corrective competence? Extending the Explanation Gate to include collaborative teach-forward protocols between humans could mitigate these social bottlenecks.

8.4 Equity, Diversity, and the AI Experience Gap

A critical area for future inquiry is the impact of Generative AI on underrepresented groups in Computer Science. Currently, Black and Hispanic students combined earn close to 20% of CS degrees in the United States [30, 34], yet they often report higher rates of imposter syndrome and lower access to early-career mentorship. Lodge and Loble [29] identify a related *metacognitive equity gap*: the cognitive risks of unstructured AI fall disproportionately on novices who lack the domain knowledge and self-regulation skills needed to use AI beneficially, compounding existing disadvantage. Generative AI has the potential to act as a leveling agent; however, if unrestricted AI usage leads to a disproportionate accumulation of epistemic debt among learners with less existing social capital, it may inadvertently widen the competence gap during technical interviews. Future research must quantify whether scaffolded AI tools like *VibeCheck* provide a more equitable pathway to senior-level architectural understanding for students from diverse socio-economic and racial backgrounds.

Artifact Availability

The *VibeCheck* VS Code/Cursor extension, the Course Scheduler experimental task suite (starter scaffold, reference implementation, 12-assertion Puppeteer grading harness, and logic bomb version), and the complete three-condition study replication protocol are openly available at https://github.com/sreecharansankaranarayanan/vibecheck

References

[1] Anthropic. 2024. Introducing Claude 3.5 Sonnet. https://www.anthropic.com/news/claude-3-5-sonnet Accessed: 2026-02-16.
[2] Anthropic. 2026. *Economic Index: March 2026 Report.* Technical Report. Anthropic. https://www.anthropic.com/research/economic-index-march-2026-report Accessed: 2026-03-26.
[3] Paris Avgeriou and Ipek Ozkaya. 2026. Panel: Technical Debt in the AI Era. International Conference on Software Engineering (ICSE). https://conf.researchr.org/info/icse-2026/panels
[4] Ryan Shaun Baker. 2005. *Designing intelligent tutors that adapt to when students game the system.* Ph. D. Dissertation. Carnegie Mellon University.
[5] Francesco Balzan, Lorenzo Angeli, Ralph Meulenbroeks, and Federica Russo. 2026. Tools or crutches? Budgeting human and machine autonomy when introducing GenAI in education. In *Artificial Intelligence in Education*, Vol. 2. Emerald Publishing Limited, Emerald Publishing Limited, Bingley, UK, 18–32.
[6] John Biggs and Kevin Collis. 1989. Towards a model of school-based curriculum development and assessment using the SOLO taxonomy. *Australian journal of education* 33, 2 (1989), 151–163.
[7] Gautam Biswas, Krittaya Leelawong, Daniel Schwartz, and Nancy Vye. 2005. Learning by teaching: A new agent paradigm for educational software. *Applied Artificial Intelligence* 19, 3-4 (2005), 363–392.

[8] Robert A Bjork. 1994. Memory management and the spacing effect. *Psychonomic Bulletin & Review* 1, 1 (1994), 1–10.

[9] Zhi-Yuan Chen, Hao Wang, Xinyu Zhang, Enrui Hu, and Yankai Lin. 2025. Beyond the surface: Measuring self-preference in llm judgments. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Miami, Florida, 1653–1672.

[10] Michelene T. H. Chi, Miriam Bassok, Matthew W. Lewis, Peter Reimann, and Robert Glaser. 1989. Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science* 13, 2 (1989), 145–182. doi:10.1016/0364-0213(89)90002-5

[11] Nicholas Choma, Sreecharan Sankaranarayanan, and R Cherukuri. 2025. Transforming expert insight into scalable AI assessment: A framework for LLM-generated metrics and user-calibrated evaluation. In *Proceedings of the IJCAI Workshop on AI for Education*. IJCAI, Montreal, Canada, 1–8.

[12] Fabrizio Dell'Acqua, Edward McFowland III, Ethan Mollick, Hila Lifshitz, Katherine C. Kellogg, Saran Rajendran, Lisa Krayer, François Candelon, and Karim R. Lakhani. 2026. Navigating the Jagged Technological Frontier: Field Experimental Evidence of the Effects of Artificial Intelligence on Knowledge Worker Productivity and Quality. *Organization Science* 37, 2 (2026), 403–423. doi:10.1287/orsc.2025.21838

[13] Paul Denny, Viraj Kumar, and Nasser Giacaman. 2023. Conversing with Copilot: Exploring prompt engineering for solving CS1 problems using natural language. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. ACM, New York, NY, USA, 1136–1142. doi:10.1145/3545945.3569823

[14] Pierre Dillenbourg. 2002. Over-scripting CSCL: The risks of blending collaborative learning with instructional design. *Three worlds of CSCL. Can we support CSCL?* 1, 1 (2002), 61–91.

[15] Yizhou Fan, Luzhen Tang, Huixiao Le, Kejie Shen, Shufang Tan, Yueying Zhao, Yuan Shen, Xinyu Li, and Dragan Gašević. 2025. Beware of metacognitive laziness: Effects of generative artificial intelligence on learning motivation, processes, and performance. *British Journal of Educational Technology* 56, 2 (2025), 489–530.

[16] Frank Fischer, Ingo Kollar, Karsten Stegmann, and Christof Wecker. 2013. *Scripting computer-supported collaborative learning: Cognitive, computational and educational perspectives*. Springer Science & Business Media, Berlin, Germany.

[17] John H. Flavell. 1979. Metacognition and cognitive monitoring: A new area of cognitive–developmental inquiry. *American Psychologist* 34, 10 (1979), 906–911. doi:10.1037/0003-066X.34.10.906

[18] Ben Hall. 2026. Epistemic Debt: The Hidden Cost of AI Speed. Failing Fast. https://failingfast.io/ai-epistemic-debt/

[19] T.B. Ionescu, S. Schlund, and C. Schmidbauer. 2020. Epistemic Debt: A Concept and Measure of Technical Ignorance in Smart Manufacturing. In *Smart Manufacturing*. Springer, Cham, Switzerland. https://link.springer.com/chapter/10.1007/978-3-030-20040-4_8

[20] Slava Kalyuga. 2009. The expertise reversal effect. In *Managing cognitive load in adaptive multimedia learning*. IGI Global Scientific Publishing, Hershey, PA, USA, 58–80.

[21] Manu Kapur. 2008. Productive failure. *Cognition and instruction* 26, 3 (2008), 379–424.

[22] Andrej Karpathy. 2023. The Hottest New Programming Language is English. Twitter/X. https://twitter.com/karpathy/status/1617979122625712128 Accessed: 2025-01-15.

[23] Andrej Karpathy. 2024. The Rise of Vibe Coding. Social Media Post (X/Twitter). Accessed: 2025-01-15.

[24] Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, et al. 2023. ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and individual differences* 103 (2023), 102274.

[25] Paul A Kirschner. 2026. Understanding AI: Offloading or Outsourcing Thinking? Paul Kirschner's Educational Blog. https://www.kirschnered.nl/2026/01/13/understanding-ai-offloading-or-outsourcing-thinking/.

[26] Nataliya Kosmyna, Eugene Hauptmann, Ye Tong Yuan, Jessica Situ, Xian-Hao Liao, Ashly Vivian Beresnitzky, Iris Braunstein, and Pattie Maes. 2025. Your Brain on ChatGPT: Accumulation of Cognitive Debt When Using an AI Assistant for Essay Writing Task. arXiv:2506.08872 [cs.HC] https://arxiv.org/abs/2506.08872 Accessed: 2026-03-26.

[27] Philippe Kruchten, Robert L Nord, and Ipek Ozkaya. 2012. Technical debt: From metaphor to theory and practice. *Ieee software* 29, 6 (2012), 18–21.

[28] Elizabeth Laird, Maddy Dwyer, and Hannah Quay de la Vallee. 2025. *Hand in Hand: Schools' Embrace of AI Connected to Increased Risks to Students*. Report. Center for Democracy & Technology. https://cdt.org/wp-content/uploads/2025/10/FINAL-CDT-2025-Hand-in-Hand-Polling-100225-accessible.pdf

[29] Jason M. Lodge and Leslie AM Loble. 2026. *Artificial Intelligence, Cognitive Offloading and Implications for Education*. Technical Report. University of Technology Sydney, Centre for Social Justice and Inclusion. doi:10.17141/4pyxmbjaq.31302475 Accessed: 2026-03-20.

[30] National Center for Education Statistics. 2023. Table 322.30. Bachelor's degrees conferred by postsecondary institutions, by race/ethnicity and field of study: Academic years 2020-21 and 2021-22. Digest of Education Statistics 2023. https://nces.ed.gov/programs/digest/d23/tables/dt23_322.30.asp Accessed: 2026-02-16.

[31] W. Quattrociocchi, V. Capraro, and M. Perc. 2025. Epistemological Fault Lines Between Human and Artificial Intelligence. arXiv:2512.19466 [cs.AI]

[32] Reuters. 2026. 'Moltbook' social media site for AI agents had big security hole, cyber firm Wiz says. Reuters Technology News. https://www.reuters.com/technology/cybersecurity/moltbook-social-media-site-ai-agents-had-big-security-hole-cyber-firm-wiz-says-2026-01-22/ Accessed: 2026-02-16.

[33] Judy Hanwen Shen and Alex Tamkin. 2026. How AI Impacts Skill Formation. arXiv:2601.20245 [cs.CY] https://arxiv.org/abs/2601.20245 Accessed: 2026-03-26.

[34] Katie Siek and Jasmine Batten. 2025. *Taulbee Survey 2024: Annual Report*. Technical Report. Computing Research Association. https://datavisualization.cra.org/TaulbeeSurvey/CRA_Taulbee_Survey_Report_2024.html Accessed: 2026-02-16.

[35] Joel Spolsky. 2002. The law of leaky abstractions. Joel on Software. https://www.joelonsoftware.com/2002/11/11/the-law-of-leaky-abstractions/

[36] John Sweller. 1988. Cognitive load during problem solving: Effects on learning. *Cognitive science* 12, 2 (1988), 257–285.

[37] Koki Wataoka, Tsubasa Takahashi, and Ryokan Ri. 2024. Self-preference bias in llm-as-a-judge. *arXiv preprint arXiv:2410.21819* abs/2410.21819, 1 (2024), 1–15.

[38] Christof Wecker and Frank Fischer. 2010. Fading Instructional Scripts: Preventing Relapses into Novice Strategies by Distributed Monitoring. In *Learning in the Disciplines: Proceedings of the 9th International Conference of the Learning Sciences (ICLS 2010) – Volume 1, Full Papers*, K. Gomez, L. Lyons, and J. Radinsky (Eds.). International Society of the Learning Sciences, Chicago, IL, 794–801. doi:10.22318/icls2010.1.794

[39] Xiaodong Wei, Lei Wang, Lap-Kei Lee, and Ruixue Liu. 2025. The effects of generative AI on collaborative problem-solving and team creativity performance in digital story creation: an experimental study. *International Journal of Educational Technology in Higher Education* 22, 1 (2025), 23.

[40] World Governments Summit. 2024. A Conversation with the Founder of NVIDIA: Who Will Shape the Future of AI? https://www.youtube.com/watch?v=8Pm2xEViNIo Interview with Jensen Huang. Quote at [00:19:00].

[41] Lixiang Yan, Viktoria Pammer-Schindler, Caitlin Mills, Andy Nguyen, and Dragan Gašević. 2025. Beyond efficiency: Empirical insights on generative AI's impact on cognition, metacognition and epistemic agency in learning. 1675–1685 pages.

[42] Lixiang Yan, Viktoria Pammer-Schindler, Caitlin Mills, Andy Nguyen, and Dragan Gašević. 2026. Beyond efficiency: Empirical insights on generative AI's impact on cognition, metacognition and epistemic agency in learning. *Computers and Education: Artificial Intelligence* 10 (2026), 100556. doi:10.1016/j.caeai.2026.100556

[43] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, Vol. 36. Curran Associates, Inc., Red Hook, NY, USA, 46595–46623.