From "Wheel-Spinning" to "Productive Failure": Research on LLM-based Hierarchical Scaffolding Mechanism in Programming Education

Xinyun Wang[1*], Bo Jiang[2],
[1] Department of Education Information Technology, East China Normal University
[2] Shanghai Institute of AI for Education, East China Normal University
[*] 51264108008@stu.ecnu.edu.cn

Abstract: *Generative AI lowers barriers to programming but risks cognitive offloading—novices outsourcing reasoning to algorithms. This study embeds an LLM-driven hierarchical tutoring mechanism into an Online Judge (OJ) system, featuring five scaffolding levels (metacognitive prompts to direct coaching) and active/passive dual-mode intervention to interrupt unproductive "wheel-spinning." A quasi-experimental study (n = 25) showed the experimental group scored significantly higher on a closed-book mid-term (M = 86.45 vs. 79.46; Cohen's d = 1.07). Process analysis revealed faster iteration cycles and a strategic "submission-as-interaction" behavior, confirmed by student interviews. We frame these results as behavioral evidence for a promising scaffolding mechanism while acknowledging confounding limitations.*

Keywords: Generative AI, Programming Education, Productive Failure, Hierarchical Scaffolding

1. Introduction

LLMs have transformed programming education: GenAI can translate natural language into executable code, eliminating syntactic friction. Yet this convenience carries a cost. When AI removes the cognitive demand of debugging and problem-solving, novices lose the opportunity to build durable mental schemas through productive struggle (Kapur, 2016). The result is a core instructional paradox—AI helps students finish tasks while undermining the very processes that build competence. This study addresses that paradox with two research questions: (RQ1) How can an LLM-based hierarchical tutoring mechanism improve programming performance? (RQ2) How does dual-mode AI intervention transform unproductive failure into productive failure?

2. Theoretical Background

Productive Failure and Wheel-Spinning. Kapur (2008, 2016) argues that exploring ill-structured problems before instruction activates prior knowledge and deepens conceptual construction, even at the cost of initial performance. In programming, however, unconstrained failure risks "wheel-spinning"—spending significant time without progress (Beck & Gong, 2013)—which can escalate into learned helplessness (Maier & Seligman, 2016). Effective design must therefore hold students within a zone of productive struggle.

Adaptive Scaffolding. Koedinger and Aleven (2007) describe the "assistance dilemma": too much help suppresses thinking; too little causes frustration. Hattie and Timperley (2007) show that feedback should prioritize self-regulation over task-level answers. Incremental, fading scaffolding is empirically superior for novice programmers (Zheng et al., 2022), with direct code hints reserved as a fail-safe worked example (Shih et al., 2010).

3. System Design

3.1 Architecture

The platform uses a B/S architecture OJ (Wasik et al., 2018) with three layers: frontend (code editor, compilation feedback), backend (LLM tutoring agent), and data persistence. For the control group, it operates as a standard OJ. For the experimental group, the backend intercepts every submission in real time, selects a scaffolding response via rule-

based routing and prompt engineering, and returns a pedagogical intervention—never raw code. Fine-grained logs capture every Run operation, error message, dwell time, and dialogue turn.

3.2 Hierarchical Scaffolding Mechanism

The system employs five scaffolding levels governed by a "reset-on-novelty" rule: a new error type resets the level to L0, granting renewed exploratory space; repeated identical errors escalate the level. This operationalizes Zheng et al.'s (2022) fading principle—maximum cognitive demand by default, degrading only when behavioral evidence demands it.

Example: an IndexError recurring for the second time triggers L1 ("Does your assumption about list length hold?"). A third occurrence escalates to L2 ("What happens when the index reaches the last element?"). A new TypeError resets to L0.

Table 1 Operational definitions of scaffolding levels

Level	Taxonomy	Instructional Description
Level 0	Baseline	No pedagogical support or feedback is provided by the system.
Level 1	Metacognitive Tutoring	The system merely prompts the student to check the alignment between their current strategy and the task requirements, promoting self-monitoring.
Level 2	Socratic Questioning	The system guides students to discover logical loopholes through reflective questioning.
Level 3	Analogical Prompting	The system utilizes similar cases or examples to inspire conceptual understanding.
Level 4	Direct Coaching	Specific code snippets or bottom-level logic corrections are provided.

3.3 Dual-Mode Interaction

When the backend detects wheel-spinning (repeated identical errors within a time window), it automatically pushes an intervention following the L1→L2→L3→L4 sequence. Clicking "Request Help" defaults to L2 (Socratic). Even if a student explicitly requests code, a pre-set system prompt forces the LLM to respond with guiding questions. Level escalation requires a secondary LLM classifier to confirm a severe knowledge gap.

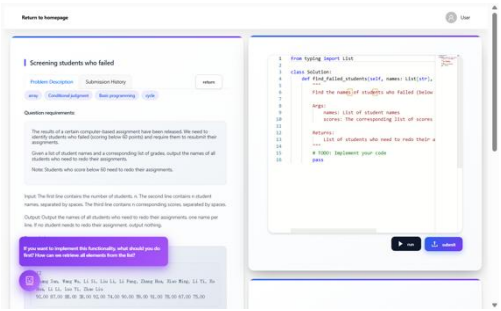

Figure 1 The active mode schematic diagram

4. Experiment

A quasi-experimental RCT was conducted in a Python programming course at a Chinese university over eight weeks (n = 25). Participants were stratified by a programming pre-test and assigned to the experimental OJ (LLM scaffolding) or standard OJ (control). Both groups completed identical weekly assignments; the same instructor delivered identical lectures. A closed-book, AI-free mid-term exam at week four served as the sole summative outcome. Platform logs recorded submission intervals, error sequences, and scaffolding interactions. Post-experiment semi-structured interviews provided qualitative context.

Three behavioral metrics were derived from logs: (1) Ineffective Attempt Rate—proportion of consecutive submission pairs with no change in error type; (2) Average Submission Interval—mean time between submissions within a one-hour session; (3) Scaffolding Level Distribution—weekly proportion of interactions at each level.

5. Results

5.1 Learning Outcomes

The experimental group (M = 86.45, SD = 5.20) significantly outperformed the control group (M = 79.46, SD = 7.40); t(23) = 2.63, p = .015, Cohen's d = 1.07. The narrower score distribution (Figure 2) suggests the mechanism particularly supported lower-performing students, reducing within-group variance and promoting equity. We note that the eight-week span introduces plausible confounds—freed from manual debugging, experimental students may have redirected time to exam-relevant study—so results should be interpreted as indicative rather than causally definitive.

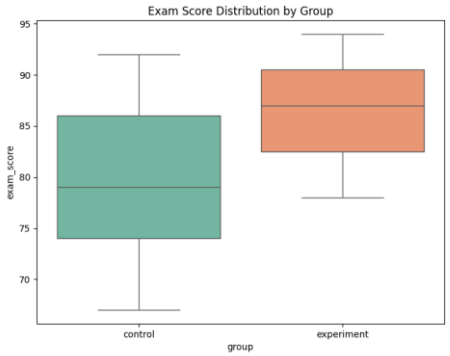

Figure 2 Comparison of mid-term exam scores between control and experimental groups

5.2 Learning Process

Both groups achieved near-100% problem completion, demonstrating equal persistence. The experimental group's average submission interval was ~120 s, one-third of the control (~360 s), consistent with a "high-frequency iteration and rapid feedback" learning loop (Figure 3). Crucially, the experimental group's superior offline exam performance rules out the explanation that frequent submissions simply offloaded assignment completion without genuine learning.

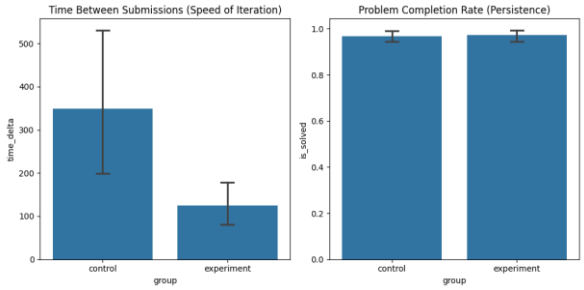

Figure 3 Comparison of submission intervals and problem completion rates

5.3 Strategic Scaffolding Use

The experimental group's Ineffective Attempt Rate was significantly higher than the control's (Figure 4)—counterintuitive under traditional definitions. Interviews revealed a deliberate "submission-as-interaction" strategy: students found submitting a known-failing snippet faster than typing a problem description in the chat interface. One participant noted: "Submitting and seeing the error is faster than explaining my confusion—the system just tells me what to look at." This reframes the high ineffective rate as active engagement with the trigger logic, not disengagement. The offline exam advantage confirms cognitive internalization rather than system exploitation.

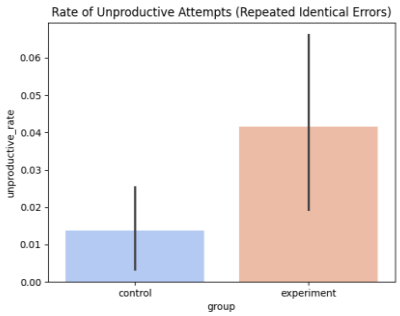

Figure 4 Comparative Analysis of Unproductive Attempt Rates (Repeated Identical Errors)

6. Limitations

Several limitations bound interpretation. The sample (n = 25, single institution) limits generalizability. A single mid-term score cannot fully capture learning; delayed retention tests are needed. Confounds over eight weeks—differential study time, potential outside-tool use in the control group, and novelty effects—cannot be fully ruled out. We position this work as behavioral evidence for a promising mechanism, not a proof of causal efficacy.

Acknowledgements

This work was partially supported by the National Natural Science Foundation of China under Grant 62477012, and the Natural Science Foundation of Shanghai under Grant 23ZR1418500.

Reference

Beck, J. E., & Gong, Y. (2013). Wheel-Spinning: Students Who Fail to Master a Skill. In H. C. Lane, K. Yacef, J. Mostow, & P. Pavlik (Eds.), *Artificial Intelligence in Education* (pp. 431–440). Springer. https://doi.org/10.1007/978-3-642-39112-5_44

Hattie, J., & Timperley, H. (2007). The Power of Feedback. *Review of Educational Research*, *77*(1), 81–112.

Kapur, M. (2008). Productive Failure. *Cognition and Instruction*, *26*(3), 379–424.

Kapur, M. (2016). Examining Productive Failure, Productive Success, Unproductive Failure, and Unproductive Success in Learning. *Educational Psychologist*, *51*(2), 289–299. https://doi.org/10.1080/00461520.2016.1155457

Koedinger, K. R., & Aleven, V. (2007). Exploring the Assistance Dilemma in Experiments with Cognitive Tutors. *Educational Psychology Review*, *19*(3), 239–264. https://doi.org/10.1007/s10648-007-9049-0

Maier, S. F., & Seligman, M. E. P. (2016). Learned Helplessness at Fifty: Insights from Neuroscience. *Psychological Review*, *123*(4), 349–367. https://doi.org/10.1037/rev0000033

Shih, B., Koedinger, K. R., & Scheines, R. (2010). A Response-Time Model for Bottom-Out Hints as Worked Examples. In *Handbook of Educational Data Mining*. CRC Press.

Wasik, S., Antczak, M., Badura, J., Laskowski, A., & Sternal, T. (2018). A Survey on Online Judge Systems and Their Applications. *ACM Comput. Surv.*, *51*(1), 3:1-3:34. https://doi.org/10.1145/3143560

Zheng, L., Zhen, Y., Niu, J., & Zhong, L. (2022). An exploratory study on fade-in versus fade-out scaffolding for novice programmers in online collaborative programming settings. *Journal of Computing in Higher Education*, *34*(2), 489–516. https://doi.org/10.1007/s12528-021-09307-w