Free or Systematic? Structured AI Guidance Improves Tutor Knowledge-State Estimation Quality and Self-Regulated Learning in Programming Education

Hanyu Zhao, Yuzhuo Wu, Zhufeng Lu, Liangyu Chen*
Shanghai Key Laboratory of Trustworthy Computing, East China Normal University
*lychen@sei.ecnu.edu.cn

Abstract

Generative AI can accelerate programming learning, but friction-free, answer-oriented help may weaken learners' self-regulation. We compare unstructured answer-oriented assistance with systematic scaffolding that provides step-by-step hints, knowledge-state feedback from a Graph-enhanced Interactive Knowledge Tracing (GIKT) model, and difficulty-matched practice. We conduct two within-subject crossover studies across Java and Python programming courses. Students alternate between conditions by concept unit with counterbalanced starting conditions. The systematic scaffolding condition delivers interactive guidance through a *Learn-Practice-Evaluate-Support* loop, providing Socratic prompts and step-by-step hints without revealing final solutions, along with personalized practice recommendations and diagnostic views. We analyze primary outcomes of self-reported transfer (Likert 1–6) and tutor knowledge-state estimation quality, quantified by Expected Calibration Error (ECE) under time-split evaluation, using linear mixed-effects models. We demonstrate that systematic scaffolding yields higher self-reported transfer with convergent trends in exam subscores, and substantially better prediction calibration (ECE: 0.13 vs. 0.24, effect size $d = 1.64$). Process analyses reveal increased planning and monitoring engagement alongside reduced time pressure and frustration, transforming engagement behaviors into productive learning processes while enhancing the diagnostic value of interaction traces. Our findings demonstrate that systematic scaffolding outperforms unstructured answer-oriented assistance, establishing it as the superior approach for AI-supported programming education.

Keywords: knowledge-state estimation, self-regulated learning, AI tutoring, cognitive load, programming education

Introduction

The integration of generative AI into programming education presents a paradox of efficiency. Large Language Models (LLMs) offer immediate, personalized solutions that reduce time-on-task, yet this friction-free access may contribute to "illusions of competence," where learners mistake the ability to retrieve a solution for the ability to construct it (Koriat and R. A. Bjork 2005; Tankelevitch et al. 2024; Awad 2025). As students increasingly rely on AI as a just-in-time answer engine, the central challenge shifts from providing access to knowledge to regulating the *process* of its acquisition.

Current research highlights a critical gap: unstructured answer-oriented assistance (e.g., direct LLM Q&A) accelerates task completion but often bypasses the cognitive struggle required for deep understanding (Giannakos and Chorianopoulos 2024; Jangra and Muresan 2025). Without pedagogical constraints, learners may engage in shallow processing patterns, and system knowledge-state estimates may become less accurate due to limited interaction data and reduced engagement depth.

To address this challenge, we investigate whether systematic scaffolding can transform surface-level efficiency into self-reported transferable competence. We compare two learning environments. (1) *Systematic Scaffolding (our system SageJavon)*, a *Learn–Practice–Evaluate–Support* loop integrating interactive guidance, practice logging, knowledge analysis, and personalized recommendations. (2) *Unstructured Answer-Oriented Assistance (using ChatGLM)* (GLM et al. 2024), providing direct answers with minimal pedagogical scaffolding. We examine how "productive friction" (Ward, Nolen, and Horn 2011; Hagel 3rd and Brown 2005; E. L. Bjork and R. A. Bjork 2011; Mynott and Zimmatore 2022), which consists of structured steps that increase process complexity while reducing extraneous cognitive burden, influences learning outcomes and tutor knowledge-state estimation quality.

We hypothesize that systematic scaffolding improves learning by slowing learners down at critical moments rather than maximizing speed. In SageJavon, we implement this approach through interactive, hint-based guidance (Silva Gonçalves 2022), continuous knowledge-state feedback via Graph-enhanced Interactive Knowledge Tracing (GIKT) (Yang et al. 2020), and adaptive practice recommendations that keep tasks within an appropriately challenging range, consistent with the zone of proximal development (ZPD) (Fani and Ghaemi 2011). The system introduces brief delays, reflection prompts, and adaptive difficulty to encourage learners to engage in the *plan* → *monitor* → *reflect* cycles central to Self-Regulated Learning (SRL) (Flavell 1979; Molenaar et al. 2023). We test this hypothesis in a crossover study ($N = 172$) spanning two programming languages. Our findings demonstrate that structured AI guidance improves both tutor knowledge-state estimation quality and self-regulated learning in programming education, establishing systematic scaffolding as the superior approach over free-form assistance.

Methodology

We compare systematic scaffolding (SageJavon) with unstructured answer-oriented assistance (ChatGLM) in a within-subject crossover design across two programming courses. This section presents our theoretical framework, research questions, experimental design, and analytical approach.

In G. Boleda, I. Dautriche, M. Maldonado, C. Saldana and T. Torrent (Eds.), *Proceedings of the 48th Annual Conference of the Cognitive Science Society*

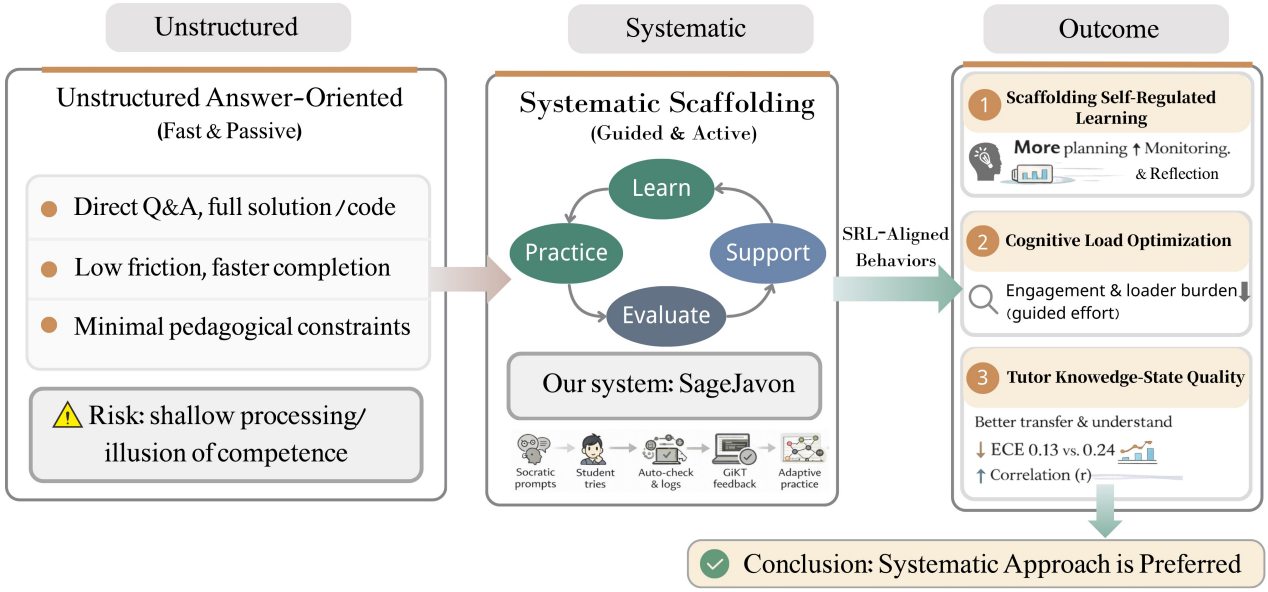


Figure 1: Comparison framework: Unstructured answer-oriented assistance vs. systematic scaffolding. Left (Intervention): Unstructured answer-oriented approach provides direct Q&A with full solutions, low friction, and faster completion, but risks shallow processing and illusion of competence. Middle (Mechanisms): Systematic scaffolding (SageJavon) implements a closed-loop *Learn–Practice–Evaluate–Support* cycle through five components: Socratic prompts, student attempts, auto-check & logs, GIKT feedback, and adaptive practice, fostering SRL-aligned behaviors. Right (Outcome): Systematic scaffolding yields (1) increased SRL-aligned behaviors (planning, monitoring, reflection), (2) cognitive load optimization (engagement with lower burden), and (3) improved learning outcomes and tutor knowledge-state quality (better transfer, lower ECE: 0.13 vs. 0.24, higher correlation).

Theoretical Framework

We integrate three perspectives to explain why systematic scaffolding should outperform unstructured answer-oriented assistance in programming education.

1. Scaffolding Self-Regulated Learning. Self-regulated learning (SRL) involves cyclical processes of planning, monitoring, and reflection (Zimmerman 2002; P. Winne 2016). Effective learners engage in metacognitive activities such as setting goals, monitoring progress, and reflecting on errors before attempting new problems. However, unstructured AI assistance can disrupt these cycles by enabling direct problem-to-solution transitions, bypassing the cognitive struggle necessary for deep learning.

Systematic scaffolding serves as an external regulator that prompts learners to complete SRL cycles. Rather than providing immediate answers, systematic scaffolding structures the learning process through heuristic questions (e.g., "*Why did this error occur?*"), step-by-step hints, and reflection prompts (Koedinger, Corbett, and Perfetti 2013; Roll, Wiese, and Aleven 2024). This structured guidance encourages learners to engage in planning (e.g., reviewing problem requirements before coding), monitoring (e.g., checking test cases and error messages), and reflection (e.g., analyzing errors before retrying). We hypothesize that these SRL-aligned behaviors mediate the relationship between systematic scaffolding and learning outcomes.

2. Cognitive Load Optimization. Cognitive Load Theory (Sweller 2011; Berssanette and Francisco 2021) separates extraneous load (irrelevant burden that impedes learning) from germane load (effort devoted to schema construction). AI support can raise extraneous load when learners struggle to prompt effectively, and reduce germane load when they copy solutions without processing the underlying concepts.

We argue that systematic scaffolding optimizes this trade-off by introducing "*productive friction*" (Ward, Nolen, and Horn 2011; E. L. Bjork and R. A. Bjork 2011): structured constraints that increase process demands (e.g., iterative attempts, stepwise hints) while reducing extraneous burden (e.g., removing prompt-engineering requirements via clear guidance). By embedding "*desirable difficulties*," scaffolding sustains engagement and supports knowledge construction (E. L. Bjork and R. A. Bjork 2011). We hypothesize that it reduces subjective workload (especially frustration and temporal pressure) without reducing engagement, producing a more efficient learning environment even with longer time-on-task.

3. Tutor Knowledge-State Estimation Quality. Personalized tutoring relies on accurate knowledge-state estimation, yet estimate quality hinges on how diagnostic learners' interaction traces are (Dunlosky and Rawson 2012). Shallow processing (e.g., directly copying AI answers) reduces trace informativeness and degrades calibration between predicted

and realized performance.

We use Graph-enhanced Interactive Knowledge Tracing (GIKT) (Yang et al. 2020) to infer learners' latent knowledge states from interaction data. GIKT represents concepts as a graph and uses interaction history to predict future performance. We hypothesize that systematic scaffolding improves calibration by eliciting more structured, diagnostic interaction patterns, thereby reducing Expected Calibration Error (ECE) (Futami and Fujisawa 2024) between predicted correctness probabilities and observed outcomes. Lower ECE indicates more reliable knowledge-state estimates and supports better adaptive tutoring decisions.

We operationalize tutor/system monitoring quality as prediction–outcome calibration, which is distinct from learners' metacognitive calibration (not directly assessed). Accordingly, we evaluate whether the *system* accurately estimates learner knowledge states, rather than learners' self-assessments of understanding.

Research Questions

Based on this theoretical framework, we address three research questions.

RQ1: Does systematic scaffolding improve learning outcomes and tutor knowledge-state estimation quality compared to unstructured answer-oriented assistance?

RQ2: Do SRL-aligned behaviors account for part of the effect of systematic scaffolding on learning outcomes?

RQ3: How does systematic scaffolding affect cognitive load and the efficiency–learning trade-off?

System Implementation (SageJavon)

SageJavon implements a closed-loop *Learn–Practice–Evaluate–Support* cycle (Figure 1). As illustrated in Figure 2, this cycle includes five components: (1) *Socratic prompts* via interactive guidance agents using Retrieval-Augmented Generation (RAG) (Gao et al. 2023), (2) *Student tries* for iterative solution attempts, (3) *Auto-check & logs* capturing behavioral traces and performance metrics, (4) *GIKT feedback* analyzing interaction data to estimate knowledge states and generate personalized knowledge graphs, and (5) *Adaptive practice* providing targeted exercises through a recommender engine. Practice recommendations target a 40%–60% predicted correctness range to maintain appropriate challenge levels.

Participants

We conduct two parallel course studies to test whether the effects of systematic scaffolding generalize across programming languages (N = 172). The Java study involves 85 students in a 12-week Object-Oriented Programming course, and the Python study involves 87 students in a 14-week Python Programming course. Across both courses, the instructional support logic is identical; adaptations are limited to language-specific syntax/semantics and corresponding test cases. The study is approved by the university's Institutional Review Board, and all participants provide informed consent.

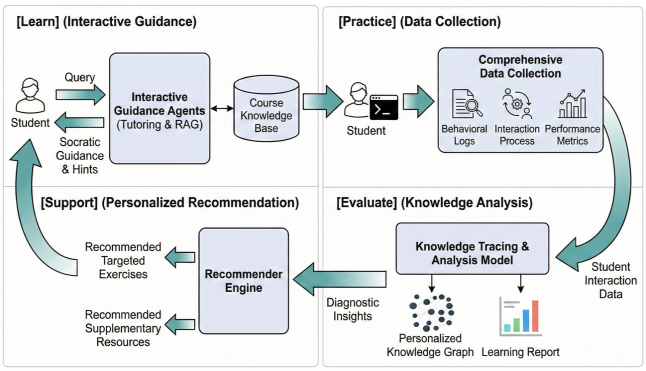


Figure 2: SageJavon system architecture showing the closed-loop *Learn–Practice–Evaluate–Support* cycle.

Experimental Design

We use a within-subject crossover design: each student experiences both support conditions. The course is divided into 10 concept units (e.g., control flow, functions, data structures, and OOP), and students alternate conditions across units. The starting condition is counterbalanced to mitigate sequence effects. Within each unit, students complete concept-aligned programming tasks using the same auto-grading pipeline. The two conditions are described below.

- **Unstructured Answer-oriented Assistance.** Students use ChatGLM as their help tool, allowing direct answers or solution snippets with minimal pedagogical scaffolding. Unlike the systematic condition, students do not receive GIKT knowledge-state feedback, personalized practice recommendations, or diagnostic views. Log analysis confirms treatment fidelity: most queries are answer-seeking (mean length: 45 words), and 78% explicitly request complete solutions or code snippets.
- **Systematic Scaffolding (SageJavon).** Students receive interactive guidance through the same underlying QA model (ChatGLM), constrained to Socratic prompts and step-by-step hints without revealing final solutions. SageJavon also provides GIKT-based knowledge-state feedback, personalized practice recommendations, and diagnostic views (personalized knowledge graphs and learning reports).

Measures

We collect data from platform interaction logs, automated code evaluation records, and standardized questionnaires. Outcome measures include: (i) self-reported knowledge transfer, measured using Likert-scale items ranging from 1 to 6 (Cronbach's α = 0.82) (Connelly 2011). (ii) final exam subscores, computed by mapping exam items to their corresponding concept units to enable within-subject comparisons; and (iii) tutor knowledge-state estimation quality, quantified using Expected Calibration Error (ECE) (Futami and Fujisawa 2024) and the correlation between predicted and observed performance.

Process Variables include three Self-Regulated Learning (SRL) indicators (P. H. Winne and Perry 2000) observable in both conditions: *Planning* (proportion of problems with material access before first submission), *Monitoring* (frequency of checking test cases/error messages per day), and *Reflection* (mean time interval between errors and retry attempts, excluding idle periods >5 minutes).

Covariates include prior programming experience, attendance, and general system engagement metrics.

Statistical Analysis

Tutor Knowledge-state Estimation Quality. We quantify tutor-side knowledge-state estimation with Expected Calibration Error (ECE) (Futami and Fujisawa 2024) under a time-split protocol. In both conditions, GIKT ingests interaction traces (submissions, attempts, correctness, and edit history) and outputs correctness probabilities. Because the conditions induce different interaction patterns, we evaluate calibration separately by condition.

For each concept unit, we train GIKT on all prior units (pooled across conditions), freeze parameters, and then predict on held-out data from the current unit for each condition (out-of-sample). We compute ECE per condition using 10 equal-width probability bins by comparing predicted probabilities to observed correctness (Futami and Fujisawa 2024). We estimate uncertainty with a block bootstrap that resamples students with replacement (10,000 iterations). Results remain stable under alternative binning (15/20 bins) and when replacing ECE with the Brier score. We also report prediction–performance correlation as the Pearson correlation between predicted probabilities and observed correctness, aggregated within each condition.

Condition Effects. We test condition effects with mixed-effects models (Luke 2017) to account for repeated observations across students and concept units. We fit linear mixed-effects models for continuous outcomes and logistic mixed-effects models for binary outcomes. Each model includes random intercepts for student and unit, and fixed effects for Condition, Period (unit index), and Order (starting condition). We probe carryover by adding a Condition × Period interaction and a PreviousCondition term in sensitivity analyses; we detect no carryover for primary outcomes ($p > 0.10$). We report p-values and Cohen's d in Results.

Multiple Comparisons. We preregister self-reported knowledge transfer and ECE as primary outcomes. We treat all other measures as secondary/exploratory, report nominal p-values, and note that conclusions do not change after Benjamini–Hochberg FDR control ($q < 0.05$) (Benjamini and Hochberg 1995).

Mediation and Growth Analyses. We test whether SRL behaviors mediate the effect of scaffolding on learning outcomes using multilevel mediation with bootstrapped confidence intervals (10,000 resamples). We compute mediators and outcomes at the unit level nested within students, and we ensure temporal ordering by using unit-level SRL measures that precede outcome measures (exam subscores). We analyze SRL trajectories with growth-curve models that include random intercepts and slopes and test Condition × Time interactions.

NASA-TLX Processing. We score NASA Task Load Index (NASA-TLX) (Hart 2006) so higher values indicate greater workload; we reverse-code Performance accordingly. We report all six dimensions. For exploratory composites, we compute (i) *Process/Interaction Burden* as temporal demand, frustration, and mental demand, and (ii) *Task Engagement* as effort and physical demand. We standardize composites across all observations to $M = 50$ and $SD = 10$.

Results

We present empirical findings from two parallel university-level programming courses (Java: $n = 85$; Python: $n = 87$; $N = 172$ total). Our crossover design enables within-subject comparisons, controlling for individual differences while examining whether systematic scaffolding effects generalize across programming languages. We organize results to address our three research questions.

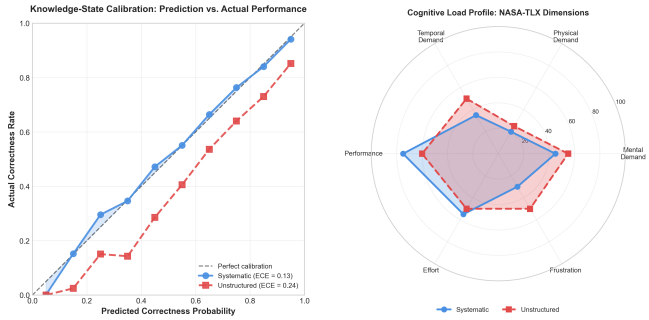
RQ1: Learning Outcomes and Tutor Knowledge-State Estimation Quality

Learning Outcomes Systematic scaffolding yields higher self-reported understanding and transfer across both programming domains (see Table 1). Linear mixed-effects models show that for *self-reported knowledge transfer*, our primary outcome measure, systematic scaffolding achieves significantly higher scores in both courses (Java: $p = 0.049$, $d = 0.43$; Python: $p = 0.042$, $d = 0.41$), representing medium effect sizes. *Self-reported concept understanding* shows similar patterns (Java: $p = 0.045$, $d = 0.42$; Python: $p = 0.038$, $d = 0.44$). Final exam subscores show convergent trends favoring systematic scaffolding, with positive correlations between self-reported transfer and exam subscores, suggesting consistency between perceived and objective measures.

Table 1: **Learning Outcomes by Course.** Self-reported knowledge transfer and concept understanding scores (Likert scale, 1–6) for Java and Python courses separately. Values are mean (SD). d = Cohen's d effect size. p-values from two-tailed tests.

Outcome	Course	Systematic	Unstructured	p	d
Knowledge Transfer	Java ($n = 85$)	5.09 (0.51)	4.82 (0.70)	0.049	0.43
	Python ($n = 87$)	5.04 (0.49)	4.79 (0.71)	0.042	0.41
Concept Understanding	Java ($n = 85$)	4.91 (0.48)	4.66 (0.65)	0.045	0.42
	Python ($n = 87$)	4.88 (0.46)	4.61 (0.68)	0.038	0.44

Tutor Knowledge-State Estimation Quality As illustrated in Figure 3 and Table 2, systematic scaffolding shows significantly stronger prediction–performance alignment compared to the unstructured condition. The prediction–performance correlation improves from $r = 0.58$ to $r = 0.72$ ($d = 0.74$). Expected Calibration Error (ECE), our primary outcome for



(a) Calibration Curve (b) Cognitive Load Profile

Figure 3: **Tutor knowledge-state estimation quality and cognitive load.** (A) GIKT calibration curves under the time-split, out-of-sample protocol ($N = 172$). Systematic scaffolding reduces ECE (0.13) relative to unstructured assistance (0.24). (B) NASA-TLX profiles (0–100; Performance reverse-coded), showing lower temporal demand and frustration with systematic scaffolding.

tutor knowledge-state estimation quality, is substantially lower in the systematic group (0.13 vs. 0.24), representing a 46% reduction ($d = 1.64$, $p < 0.001$). The improved calibration reflects enhanced interaction trace diagnosticity and deeper learning engagement.

Answer to RQ1: Systematic scaffolding significantly improves both learning outcomes (higher self-reported transfer and concept understanding, $d = 0.41$–0.44) and tutor knowledge-state estimation quality (46% reduction in ECE, $d = 1.64$), demonstrating superior performance across both dimensions compared to unstructured answer-oriented assistance.

RQ2: SRL-Aligned Behaviors as Mediators

To explain why systematic scaffolding improves learning outcomes, we examine whether SRL-aligned behaviors mediate this effect.

SRL Behavioral Mediation We test whether SRL behaviors mediate the effect of systematic scaffolding on exam scores. We fit a multilevel mediation model with unit-level mediators and outcomes nested within students and derive bootstrap confidence intervals (10,000 resamples). To preserve temporal ordering, we compute mediators at the unit level before the outcome (exam subscores).

We operationalize *resource engagement* as per-unit accesses to course materials, reviews of problem requirements, and test-case checks. We operationalize *knowledge monitoring* as per-unit reviews of error messages and code-correctness checks. Systematic use shows significant indirect effects on exam scores via resource engagement (0.23, 95% CI [0.12, 0.35], $p < 0.001$) and knowledge monitoring (0.18, 95% CI [0.08, 0.29], $p = 0.002$). The direct effect remains significant after adding mediators ($\beta = 0.31$, $p = 0.008$), indicating partial mediation. Together, these results support a behavioral pathway: scaffolded workflows increase engagement and monitoring actions that predict better outcomes.

Table 2: Tutor knowledge-state estimation quality. Expected Calibration Error (ECE) and prediction–performance correlation (r) pooled across Java and Python ($N = 172$), mean (SD). Diff. = Systematic − Unstructured. Lower ECE and higher r indicate better estimation; we find no Course×Condition interaction ($p > 0.20$).

Measure	Systematic	Unstructured	Diff.	p	d
ECE (lower is better)	0.13 (0.05)	0.24 (0.08)	-0.11	< 0.001	1.64
Monitoring accuracy (r)	0.72 (0.15)	0.58 (0.22)	+0.14	0.001	0.74

Table 3: Cognitive load composites. Standardized NASA-TLX composite scores ($M = 50$, $SD = 10$) pooled across Java and Python ($N = 172$), mean (SD). Diff. = Systematic − Unstructured. Process/Interaction Burden aggregates temporal demand, frustration, and mental demand; Task Engagement aggregates effort and physical demand.

Composite	Systematic	Unstructured	Diff.	p	d
Process/Interaction Burden	37.42 (8.2)	43.05 (7.8)	-5.63	< 0.001	0.70
Task Engagement	62.90 (9.1)	62.44 (8.9)	+0.46	0.042	0.05

Temporal Growth of SRL Weekly trajectories show that systematic scaffolding increasingly promotes SRL-aligned behaviors over time (Table 4; Figure 4). We track behaviorally observable indicators in both conditions (material access, test-case and error-message checks, error-to-retry intervals), noting that the systematic condition also offers additional monitoring affordances (knowledge graphs, learning reports, GIKT feedback). Growth-curve models yield significant Condition×Time interactions for all SRL indicators ($p < 0.001$), indicating that the advantage accumulates across the course. The SRL composite grows 2.7× faster under systematic scaffolding, consistent with a positive feedback process in which early support for planning, monitoring, and reflection elicits more SRL-aligned actions over time.

Answer to RQ2: SRL-aligned behaviors (resource engagement and knowledge monitoring) partially mediate the effect of systematic scaffolding on learning outcomes, with significant indirect effects ($p < 0.001$ and $p = 0.002$, respectively) and a 2.7× higher growth rate in SRL behaviors over time, supporting the behavioral pathway explanation.

RQ3: Cognitive Load and the Efficiency–Learning Trade-off

Cognitive Load Optimization Systematic scaffolding reduces subjective workload burden, particularly in process efficiency dimensions (Table 3; Figure 3). Linear mixed-effects models show that *process/interaction burden composite* is significantly reduced in the systematic condition (13% reduction, $d = 0.70$, $p < 0.001$), while *task engagement composite* shows

Table 4: SRL trajectories over the term. Model-implied weekly means of SRL indicators by condition (Sys.=systematic scaffolding; Unstr.=unstructured). Planning: proportion of problems with material access before the first submission (0–1). Monitoring: mean number of test-case or error-message checks per active day. Reflection: mean post-error pause before the next attempt (min). SRL composite: z-scored unit-level composite aggregated by week. Slope: estimated weekly change from the growth model (units match each indicator).

Time point	**Planning**		**Monitoring (/day)**		**Reflection (min)**		**SRL (composite)**	
	Sys.	**Unstr.**	**Sys.**	**Unstr.**	**Sys.**	**Unstr.**	**Sys.**	**Unstr.**
Week 2	0.42	0.28	2.30	1.10	8.50	4.20	0.38	0.24
Week 4	0.51	0.31	3.10	1.40	12.30	5.10	0.45	0.27
Week 6	0.58	0.33	3.80	1.60	15.60	5.80	0.52	0.29
Week 8	0.62	0.35	4.20	1.70	18.20	6.10	0.56	0.31
Week 10	0.65	0.36	4.50	1.80	19.80	6.30	0.59	0.32
Week 12	0.68	0.37	4.70	1.90	21.40	6.50	0.61	0.33
Slope	**0.023**	**0.007**	**0.20**	**0.07**	**1.08**	**0.19**	**0.019**	**0.007**

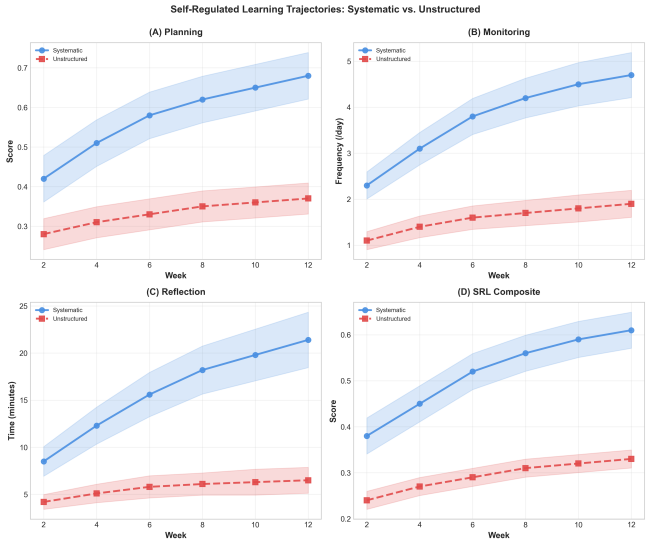


Figure 4: Self-regulated learning trajectories. Weekly SRL indicators (Planning, Monitoring, Reflection, and Composite) show higher growth under systematic scaffolding (solid circles) than unstructured assistance (dashed squares). Shading denotes 95% CIs. Condition differences in growth rates are significant ($p < 0.001$).

a statistically significant but practically negligible increase ($d = 0.05$, $p = 0.042$). The "productive friction" in systematic scaffolding increases process complexity but reduces subjective frustration and temporal pressure, creating a more efficient learning environment despite longer time-on-task.

The Efficiency–Learning Trade-off We examine the trade-off between surface-level efficiency and learning outcomes. In the unstructured condition, speed-oriented signals (lower perceived time pressure, minimal help-seeking) show weak or negative associations with learning outcomes, suggesting that being faster does not reliably translate into better learning. In contrast, systematic scaffolding elicits more engagement behaviors (help-seeking, monitoring interactions, longer time on task) that align positively with learning outcomes. Systematic scaffolding produces the largest gains on tutor-side monitoring quality (ECE reduction, $d = 1.64$) and shows positive effects on monitoring accuracy and downstream learning outcomes. These patterns align with the Cognitive (In)Efficiency account that systematic scaffolding turns engagement into productive learning while unstructured assistance decouples speed from learning.

Answer to RQ3: Systematic scaffolding optimizes cognitive load by reducing process/interaction burden by 13% ($d = 0.70$, $p < 0.001$) while maintaining task engagement, and transforms engagement behaviors into productive learning outcomes, demonstrating that productive friction creates a more efficient learning environment despite longer time-on-task.

Conclusion

We address the question *Free or Systematic?* through two within-subject crossover studies with 172 students in Java and Python programming courses. Results show that systematic scaffolding outperforms unstructured answer-oriented assistance, yielding higher learning outcomes ($d = 0.41$–0.44), better tutor knowledge-state estimation quality (46% ECE reduction, $d = 1.64$), improved self-regulated learning behaviors, and more adaptive cognitive load. By operationalizing "productive friction" through a closed-loop *Learn–Practice–Evaluate–Support* design with Socratic hints, GIKT-based feedback, and difficulty-matched practice, this work shows how structured engagement can improve learning processes, interaction-based diagnosis, and the design of AI tutors for programming education.

Acknowledgements

This work was supported by NSFC (No. 62272416), National Education Sciences Planning Project (No. BCA240048).

References

Awad, Omar (2025). "Deep fake learning: AI's role in misinformation and illusion of expertise." In: *Available at SSRN 5174838*.

Benjamini, Yoav and Yosef Hochberg (1995). "Controlling the false discovery rate: a practical and powerful approach to multiple testing." In: *Journal of the Royal Statistical Society: Series B (Methodological)* 57.1, pp. 289–300.

Berssanette, João Henrique and Antonio Carlos de Francisco (2021). "Cognitive load theory in the context of teaching and learning computer programming: a systematic literature review." In: *IEEE Transactions on Education* 65.3, pp. 440–449.

Bjork, Elizabeth Ligon and Robert A. Bjork (2011). "Making things hard on yourself, but in a good way: creating desirable difficulties to enhance learning." In: *Psychology and the Real World: Essays Illustrating Fundamental Contributions to Society*, pp. 56–64.

Connelly, Lynne M (2011). "Cronbach's alpha." In: *Medsurg Nursing* 20.1, pp. 45–47.

Dunlosky, John and Katherine A. Rawson (2012). "Overconfidence produces underachievement: inaccurate self evaluations undermine students' learning and retention." In: *Learning and Instruction* 22.4, pp. 271–280.

Fani, Tayebeh and Farid Ghaemi (2011). "Implications of Vygotsky's zone of proximal development (ZPD) in teacher education: ZPTD and self-scaffolding." In: *Procedia-Social and Behavioral Sciences* 29, pp. 1549–1554.

Flavell, John H. (1979). "Metacognition and cognitive monitoring: a new area of cognitive–developmental inquiry." In: *American Psychologist* 34.10, pp. 906–911.

Futami, Futoshi and Masahiro Fujisawa (2024). "Information-theoretic generalization analysis for expected calibration error." In: *Advances in Neural Information Processing Systems* 37, pp. 84246–84297.

Gao, Yunfan et al. (2023). "Retrieval-augmented generation for large language models: a survey." In: *arXiv preprint arXiv:2312.10997* 2.1.

Giannakos, Michail and Konstantinos Chorianopoulos (2024). "The promise and challenges of generative AI in education." In: *Education And Information Technologies* 29, pp. 2345–2368.

GLM, Team et al. (2024). "Chatglm: a family of large language models from glm-130b to glm-4 all tools." In: *arXiv preprint arXiv:2406.12793*.

Hagel 3rd, John and John Seely Brown (2005). "Productive friction: how difficult business partnerships can accelerate innovation." In: *Harvard Business Review* 83.2, pp. 82–91.

Hart, Sandra G (2006). "NASA-task load index (NASA-TLX); 20 years later." In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. Vol. 50. 9. Sage publications Sage CA: Los Angeles, CA, pp. 904–908.

Jangra, Anubhav and Smaranda Muresan (2025). "Designing and evaluating hint generation systems for science education." In: *arXiv preprint arXiv:2510.21087*.

Koedinger, Kenneth R., Albert T. Corbett, and Charles Perfetti (2013). "The knowledge-learning-instruction framework: bridging the science-practice chasm to enhance robust student learning." In: *Cognitive Science* 37.5, pp. 757–798.

Koriat, Asher and Robert A. Bjork (2005). "Illusions of competence in monitoring one's knowledge during study." In: *Journal of Experimental Psychology: Learning, Memory, and Cognition* 31.2, pp. 187–194.

Luke, Steven G (2017). "Evaluating significance in linear mixed-effects models in R." In: *Behavior Research Methods* 49.4, pp. 1494–1502.

Molenaar, Inge et al. (2023). "Measuring self-regulated learning and the role of AI: five years of research using multimodal multichannel data." In: *Computers in Human Behavior* 139, p. 107540.

Mynott, John Paul and Michaela Zimmatore (2022). "Pracademic productive friction: boundary crossing and pressure points." In: *Journal of Professional Capital and Community* 7.1, pp. 45–56.

Roll, Ido, Eliane S. Wiese, and Vincent Aleven (2024). "Metacognitive support in intelligent tutoring systems: when and how it helps." In: *Proceedings of the Annual Conference of the Cognitive Science Society*. Vol. 46. Toronto, Canada: Cognitive Science Society, pp. 1234–1240.

Silva Gonçalves, Jorge Alexandre da (2022). "A hint generation system for introductory programming exercises in Java." MA thesis. ISCTE-Instituto Universitario de Lisboa (Portugal).

Sweller, John (2011). "Cognitive load theory." In: *Psychology of Learning and Motivation*. Ed. by Jose P. Mestre and Brian H. Ross. Vol. 55. San Diego, CA: Academic Press, pp. 37–76.

Tankelevitch, Leonid et al. (2024). "The metacognitive demands and opportunities of generative AI." In: *ACM Transactions on Computer-Human Interaction*. Highly cited perspective on metacognition and AI interaction.

Ward, Christopher J, Susan B Nolen, and Ilana S Horn (2011). "Productive friction: how conflict in student teaching creates opportunities for learning at the boundary." In: *International Journal of Educational Research* 50.1, pp. 14–20.

Winne, Philip (2016). "Self-regulated learning." In: *SFU Educational Review* 9.

Winne, Philip H and Nancy E Perry (2000). "Measuring self-regulated learning." In: *Handbook of Self-Regulation*. Elsevier, pp. 531–566.

Yang, Yang et al. (2020). "GIKT: a graph-based interaction model for knowledge tracing." In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pp. 299–315.

Zimmerman, Barry J (2002). "Becoming a self-regulated learner: an overview." In: *Theory into Practice* 41.2, pp. 64–70.