# Modeling Learner-AI Interaction in Time-Constrained Programming Tasks

Omkar Joshi
Centre for Educational Technology, Indian Institute of Technology Bombay, Mumbai, India
omkar.joshi21@gmail.com

## ABSTRACT

The increasing use of AI coding assistants in programming environments is transforming how learners approach problem solving, particularly under time-constrained conditions. While prior research has examined perceptions and performance outcomes, there is limited understanding of how learners interact with AI tools at a behavioral level. This doctoral research adopts a data-driven approach to model learner-AI interaction in programming tasks, grounded in the theoretical framework of Self-Regulated Learning (SRL). The study collects fine-grained interaction logs from undergraduate and early-graduate computer science students completing algorithmic programming tasks with an AI coding assistant under both timed and untimed conditions. By capturing sequences of learner actions and AI queries, the research aims to (1) characterize AI usage patterns using process mining and sequential analysis, (2) model how time pressure alters behavioral transitions between coding, debugging, and AI-assisted activities, (3) identify distinct learner profiles through clustering of interaction features, and (4) develop and validate behavioral proxies for cognitive engagement and help-seeking using convergent analysis with self-report measures and qualitative examination of interaction sessions. Preliminary studies indicate a shift toward outcome-oriented behaviors and varied reliance on AI under time pressure, motivating the need for systematic modeling of interaction processes. The expected contributions include new representations of learner-AI interaction, validated methods for operationalizing learning constructs in AI-mediated environments, and empirical insights into how time constraints shape AI-assisted programming behavior.

## Keywords

AI-assisted programming, learner behavior modeling, cognitive engagement, time-constrained programming

## 1. INTRODUCTION

Omkar Joshi. Modeling Learner-AI Interaction in Time-Constrained Programming Tasks. In Anthony Botelho, Maria Mercedes T. Rodrigo, Adish Singla, Hiroaki Ogata, Hyojeong So, and Young Hoan Cho (eds.) Proceedings of the 19th International Conference on Educational Data Mining, Seoul, Republic of Korea, June, 2026, pp. 870–874. International Educational Data Mining Society (2026).


https://doi.org/10.5281/zenodo.21039967

The increasing integration of generative artificial intelligence (GenAI) tools into programming environments is transforming how learners approach problem solving. AI coding assistants now provide real-time support for code generation, debugging, and explanation, enabling learners to complete tasks more efficiently [4, 14]. While these tools offer clear productivity benefits, their impact on learning processes remains insufficiently understood, particularly in time constrained contexts such as competitive programming, technical interviews, and timed assessments, where learners must balance speed, correctness, and depth of understanding under pressure. Time pressure interacts with AI availability in ways that may fundamentally alter learner behavior. Preliminary evidence suggests that under time constraints, learners shift toward outcome-oriented strategies, using AI to generate or regenerate complete solutions rather than engaging in systematic reasoning [9]. However, these observations are based primarily on self-reports and outcome-based measures. The fine-grained behavioral dynamics of how learners integrate AI assistance into their problem-solving processes - when they seek help, how they transition between independent work and AI-assisted activity, and how they regulate these decisions, remain largely unexamined. This gap is particularly significant for the Educational Data Mining (EDM) community, where understanding learning processes through behavioral data is a central concern [2]. This doctoral research addresses this gap by developing a data-driven approach to modeling learner-AI interaction in programming tasks, grounded in the framework of SRL [16]. The study collects fine-grained interaction logs from computer science students completing algorithmic programming tasks with an AI coding assistant under both timed and untimed conditions. Using process mining, sequential analysis, and clustering techniques, the research aims to characterize AI usage patterns, model temporal dynamics of assistance-seeking, identify distinct learner profiles, and develop validated behavioral proxies for constructs such as cognitive engagement and help-seeking.

## 2. RELATED WORK

### 2.1 Learning Analytics in Programming

EDM research has extensively used fine-grained interaction logs, code edits, compilations, submissions, and error events to model learner behavior in programming environments. Techniques such as sequence mining, process modeling, and clustering have been applied to identify problem solving strategies, detect learning patterns, and predict performance [6, 10, 12, 13]. These studies demonstrate the value of be-

havioral data in uncovering latent learning processes beyond final outcomes. However, this body of work assumes environments where learners interact primarily with compilers, editors, and instructional resources. The emergence of AI coding assistants introduces a fundamentally different form of interaction, where learners can offload core cognitive tasks such as code generation and debugging to an AI system, necessitating new analytical approaches that account for AI-mediated actions within behavioral sequences.

## 2.2 AI-Assisted Programming and Learner Behavior

Recent studies have examined AI coding assistants in programming education, finding that such tools can reduce task completion time and lower barriers to entry, but may also encourage over-reliance, superficial engagement, and reduced opportunities for deep learning [4, 11, 14]. Learners frequently use AI to generate complete solutions or iterate through alternatives, especially under performance pressure [9]. While valuable, these findings are largely based on self-reports, controlled experiments, or outcome-based measures. Comparatively little work captures and analyzes fine-grained interaction traces between learners and AI systems during programming tasks, leaving the micro-level dynamics of how learners integrate AI assistance into their problem-solving processes insufficiently understood.

A growing body of EDM research has begun addressing learning in AI-mediated environments more broadly, emphasizing challenges in representing complex interaction sequences, distinguishing productive from unproductive AI usage, and operationalizing constructs such as engagement and self-regulation from behavioral data [3, 15]. However, these efforts have not specifically examined learner-AI interaction in time-constrained programming contexts, where the interplay between time pressure, AI availability, and learner strategies may fundamentally alter regulatory behavior.

## 2.3 Research Gap

Prior work has (i) established log-based behavioral analysis as a productive approach to studying programming learning, and (ii) identified the growing influence of AI assistants on learner behavior. However, a critical gap remains in the data-driven modeling of learner-AI interaction sequences, particularly in contexts where time constraints shape decision-making and strategy use. Existing studies do not provide representations that integrate Integrated Development Environment (IDE) actions and AI interactions into unified behavioral traces, nor do they offer validated operationalizations of learning constructs such as cognitive engagement and help-seeking in AI-mediated environments. This research addresses this gap by applying sequential and process mining methods to model learner-AI interaction under both timed and untimed conditions, identifying distinct learner profiles, and developing behavioral proxies grounded in the SRL framework described in Section 4.

# 3. RESEARCH OBJECTIVES

The research is guided by four objectives, organized progressively from behavioral characterization to construct operationalization.

**Objective 1: Characterize learner-AI interaction behaviors using log data** — This objective focuses on identifying how learners integrate AI assistance into their programming workflows. By analyzing interaction logs, the study aims to uncover patterns in AI usage, including frequency, timing, and the sequencing of actions such as code edits, compilation, and AI queries. This objective provides the descriptive foundation upon which subsequent analyses are built.

**Objective 2: Model temporal and sequential patterns of AI-assisted problem solving** — Building on the behavioral characterization from Objective 1, this objective applies process mining and sequential modeling techniques to understand how learners transition between activities such as coding, debugging, and seeking AI assistance, and how these transitions differ under timed versus untimed conditions. The goal is to identify recurring behavioral structures and to examine how time pressure reshapes the dynamics of AI-assisted problem solving.

**Objective 3: Identify distinct learner profiles based on AI usage strategies** — Learners may differ significantly in how they use AI assistance, ranging from minimal reliance to heavy dependence. This objective aims to group learners into meaningful profiles based on features derived from interaction sequences and aggregated metrics, enabling a comparative understanding of diverse approaches to AI-assisted problem solving and their associations with task performance.

**Objective 4: Develop and validate data-driven proxies for learning-related constructs** — Direct measurement of constructs such as cognitive engagement and help-seeking is challenging in programming contexts. This objective focuses on operationalizing such constructs using observable behavioral indicators derived from interaction data, and validating these proxies through convergent analysis with self-report measures and qualitative examination of selected interaction sessions. This objective represents the primary conceptual contribution of the research, bridging learning science constructs with computational analysis of learner-AI interaction.

Together, these objectives form a progressive arc from empirical groundwork (Objectives 1-2) through learner variation (Objective 3) to theoretically grounded construct validation (Objective 4).

# 4. THEORETICAL FRAMING

This research is situated within the theoretical framework of SRL, which describes how learners plan, monitor, and adapt their strategies during problem solving [16]. SRL provides an integrative lens that encompasses two constructs central to this work: cognitive engagement and help-seeking behavior. Rather than treating these as independent constructs, this research models them as observable manifestations of self-regulatory processes within AI-mediated programming environments. Cognitive engagement , the extent to which learners invest mental effort in understanding problems [5], is typically reflected in iterative refinement and systematic debugging in programming contexts. However, AI assistance may shift effort distribution toward faster but less effortful strategies. From an SRL perspective, engagement reflects the learner's monitoring and control processes: whether they

actively evaluate their understanding or passively accept AI-generated outputs. Help-seeking, a key SRL strategy [1], reflects when and how learners seek external support. Prior research distinguishes adaptive help-seeking (targeted and strategic) from maladaptive patterns such as over-reliance or avoidance [1]. In AI-mediated environments, AI queries represent a form of help-seeking whose cost and accessibility differ fundamentally from traditional sources such as instructors or peers, potentially altering regulatory decisions, particularly under time pressure, where learners must balance speed benefits of AI assistance against learning benefits of independent effort. Time pressure introduces an additional regulatory demand that interacts with both constructs. In time-constrained settings, learners must continuously allocate limited time across competing activities, e.g. coding, debugging, testing, and seeking AI assistance, making SRL processes particularly visible through behavioral traces. The temporal distribution of these activities, and the transitions between them, provide a window into how learners regulate their problem-solving process under constraint. This integrated framing directly informs the methodological approach adopted in this research. Specifically, SRL theory motivates the use of sequence analysis to capture regulatory transitions between activities (Objective 2), the operationalization of cognitive engagement and help-seeking through behavioral indicators rather than self-report alone (Objective 4), and the identification of learner profiles that reflect distinct regulatory strategies (Objective 3). By grounding computational representations in SRL theory, this work bridges learning science constructs with data-driven analysis, enabling interpretable modeling of learner-AI interaction in time-constrained programming tasks.

# 5. METHODOLOGY

This research adopts a data-driven, mixed-methods approach to model learner-AI interaction behaviors in programming environments. The methodology focuses on collecting fine-grained interaction data and applying analytical techniques to uncover behavioral patterns, temporal dynamics, and learning relevant indicators.

## 5.1 Data Sources and Instrumentation

Data will be collected from undergraduate and early-graduate computer science students engaged in timed programming tasks at an Indian technical university. Participants will be recruited from students with prior coursework in programming and familiarity with AI coding assistants, ensuring a baseline level of competence. A target sample of approximately 60-80 participants is planned across multiple data collection sessions to support meaningful clustering and sequence analyses. Participants will complete a set of algorithmic programming tasks in Python or C++ within fixed time limits (30-45 minutes per task), designed to reflect the structure and difficulty of competitive programming and timed assessment scenarios. Tasks will span multiple difficulty levels to elicit variation in problem-solving strategies and AI usage. To examine the role of time pressure, a subset of sessions will include a comparison condition where participants complete equivalent tasks without strict time constraints, enabling analysis of how time pressure shapes interaction behaviors. The programming environment will be instrumented using a combination of IDE-level logging and AI interaction capture. Participants will use a standard code editor (e.g., VS Code) integrated with an AI coding assistant (e.g., GitHub Copilot or a GPT-based interface), configured to log all interactions. The primary data sources include: (i) IDE Interaction Logs capturing code edits, compilation attempts, execution events, error occurrences, and associated timestamps; (ii) AI Interaction Logs recording prompts submitted by learners, AI-generated responses, acceptance or rejection of suggestions, and the timing of each exchange; (iii) Task Performance Data including task completion status, correctness of solutions, time-to-completion, and number of submission attempts; and (iv) Self-Report Measures administered before and after sessions, including survey instruments for perceived cognitive load, AI reliance, and self-regulated learning strategies, used both for triangulation and for validating behavioral proxies developed under Objective 4.

## 5.2 Data Representation

Learner interactions will be represented as time-stamped sequences of discrete events, integrating IDE actions and AI interactions into a unified trace. Each learner session will be modeled as an ordered sequence of state transitions (e.g., edit $\rightarrow$ compile $\rightarrow$ error $\rightarrow$ AI query $\rightarrow$ edit $\rightarrow$ submit), preserving both the order and timing of actions. These sequences serve as the primary input for the process mining and Markov modeling described in Stage 1 of the analysis. In addition, higher-level features will be extracted from these sequences to support the clustering and profiling in Stage 2, including transition entropy (capturing regularity of behavior), dominant n-grams (capturing recurring micro-strategies), temporal features (such as mean interaction intervals and time allocation across activity types), and AI dependency ratios (such as the proportion of debugging episodes that involve AI queries versus independent resolution).

## 5.3 Data Analysis

The analytical approach is organized around three complementary stages, each aligned with specific research objectives.

*Stage 1: Sequence and Process Analysis (Objectives 1 and 2)* — Learner interaction sequences will be analyzed using process mining techniques, specifically Directly-Follows Graphs and fuzzy mining algorithms implemented in tools such as ProM or PM4Py, to visualize and compare dominant behavioral pathways across learners [3]. Additionally, first-order Markov models will be fitted to learner action sequences to estimate transition probabilities between states such as coding, compiling, debugging, and AI querying. These transition matrices will serve as compact representations of individual problem-solving processes and will be compared across timed and untimed conditions to examine how time pressure alters behavioral transitions.

*Stage 2: Clustering and Learner Profiling (Objective 3)* — Feature vectors will be constructed for each learner session, combining sequence-derived features (e.g., transition entropy, most frequent n-grams, proportion of time in each activity state) with aggregated interaction metrics (e.g., total AI queries, mean time between AI requests, ratio of AI-assisted to independent code edits). These feature vectors will be used as input to clustering algorithms. Agglomerative hierarchical clustering will be applied initially to ex-

plore the natural grouping structure, followed by k-medoids clustering for robustness to outliers. Cluster validity will be assessed using silhouette scores and domain-informed interpretation. The resulting profiles will be characterized in terms of their AI reliance patterns, temporal strategies, and association with task performance outcomes.

*Stage 3: Construct Operationalization and Validation (Objective 4)* — Behavioral proxies for cognitive engagement and help-seeking will be derived from interaction data. Cognitive engagement will be operationalized through indicators such as the duration of sustained independent coding episodes, the ratio of self-initiated to AI-delegated debugging, and the presence of iterative refinement cycles. Help-seeking will be operationalized through the timing of AI queries relative to task progression, the specificity of prompts (classified through manual coding of a subsample), and the dependency of subsequent actions on AI outputs (e.g., accepted verbatim, modified, or discarded). These proxies will be validated through convergent analysis with self-report measures and stimulated recall interviews with a subset of participants.

# 6. WORK DONE SO FAR

The proposed research builds on three preliminary studies. First, a survey-based study of learner perceptions [7] found that learners view AI tools as performance enhancers while expressing concerns about over-reliance, and that they differentiate between practice and evaluative contexts, directly motivating the inclusion of varied task conditions in the proposed design (Section 5.1). Second, an empirical study of AI interaction behaviors during programming tasks [9] revealed outcome-oriented strategies and iterative AI-dependent debugging under time pressure, providing the behavioral evidence that motivates the sequence analysis under Objectives 1 and 2, and generating preliminary interaction traces that will serve as pilot data for the analytical pipeline. Third, an analysis of competitive programming workflows [8] identified fragmented toolchains as a barrier to capturing coherent behavioral data, directly informing the decision to use a controlled, integrated programming environment rather than observing naturalistic but fragmented settings (Section 5.1). Together, these studies provide the empirical foundation , from perceptions through observed behaviors to environmental constraints, upon which the proposed modeling builds.

# 7. EXPECTED CONTRIBUTIONS TO EDM

This research aims to contribute to the EDM community across methodological, empirical, and conceptual dimensions.

1. **Unified representations of learner-AI interaction** — Existing learning analytics approaches in programming typically model learner interactions with compilers and editors, treating the programming environment as a closed system. This work contributes new representational approaches that integrate AI interactions, prompts, responses, acceptance and rejection of suggestions into unified behavioral traces alongside IDE actions. These representations extend the analytical vocabulary available to EDM researchers studying any environment where learners interact with intelligent systems, not only programming.

2. **Validated behavioral proxies for learning constructs in AI-mediated contexts** — A persistent challenge in EDM is operationalizing theoretical constructs through behavioral data. This research contributes proxies for cognitive engagement and help-seeking that are specifically designed for AI-mediated environments and validated through convergent analysis with self-report instruments and qualitative examination via stimulated recall. By grounding these proxies in the SRL framework and demonstrating their validity, the work provides a reusable methodological template for researchers seeking to measure learning-relevant constructs in settings where AI tools are embedded in the learning process.

3. **Empirical evidence on the role of time pressure in learner-AI interaction** — By comparing interaction behaviors under timed and untimed conditions, this research provides the first controlled empirical evidence of how time constraints shape AI usage patterns, behavioral transitions, and learner regulatory strategies in programming tasks. These findings contribute to a broader understanding of learning under performance pressure and have direct implications for the design of timed assessments in AI-enabled environments, where the interaction between time constraints and AI availability raises unresolved questions about what such assessments actually measure.

4. **Learner profiles for AI-assisted programming** — The identification of distinct profiles based on interaction patterns and AI usage strategies provides a basis for differentiated instructional approaches, enabling researchers and system designers to respond to meaningful variation in how learners integrate AI assistance.

# 8. FEEDBACK SOUGHT

This doctoral research seeks feedback from the EDM community on four aspects of the proposed work. (1) Sequential modeling choices: whether first-order Markov models and process mining are sufficient to capture learner-AI interaction complexity, or whether higher-order models (e.g., hidden Markov models) or alternative representations (e.g., epistemic network analysis) would be more appropriate, and how to balance model complexity against interpretability. (2) Construct validation: the adequacy of the proposed convergent validation design combining self-report measures and stimulated recall interviews, including guidance on sample size for stimulated recall and whether additional validation sources (e.g., think-aloud protocols, expert coding of traces) should be considered. (3) Productive versus unproductive AI usage: how to distinguish strategic AI use from dependency under time pressure using behavioral features beyond frequency alone, such as prompt specificity, degree of modification of AI output, and timing relative to independent attempts. (4) Generalizability: strategies for assessing whether findings from a controlled programming environment transfer to naturalistic contexts such as coursework and competitive programming.

## 9. REFERENCES


[1] V. Aleven, E. Stahl, S. Schworm, F. Fischer, and R. Wallace. Help seeking and help design in interactive learning environments. *Review of educational research*, 73(3):277–320, 2003.

[2] R. S. Baker, K. Yacef, et al. The state of educational data mining in 2009: A review and future visions. *Journal of educational data mining*, 1(1):3–17, 2009.

[3] M. Bannert, P. Reimann, and C. Sonnenberg. Process mining techniques for analysing patterns and strategies in students' self-regulated learning. *Metacognition and learning*, 9(2):161–185, 2014.

[4] S. Barke, M. B. James, and N. Polikarpova. Grounded copilot: How programmers interact with code-generating models. *Proceedings of the ACM on Programming Languages*, 7(OOPSLA1):85–111, 2023.

[5] J. A. Fredricks, P. C. Blumenfeld, and A. H. Paris. School engagement: Potential of the concept, state of the evidence. *Review of educational research*, 74(1):59–109, 2004.

[6] M. C. Jadud. Methods and tools for exploring novice compilation behaviour. In *Proceedings of the second international workshop on Computing education research*, pages 73–84, 2006.

[7] O. Joshi and S. Das. Generative ai in coding: Elevating efficiency or eroding skills? insights from an indian user study. In *Proceedings of the 15th International Learning Analytics and Knowledge Conference Companion (LAK '25 Companion)*, pages 200–202. Society for Learning Analytics Research (SoLAR), Mar. 2025. Companion Proceedings.

[8] O. Joshi, K. Mehta, S. Kashyap, A. Purohit, and S. Das. Compete and learn? understanding competitive programming ide workflows and designing context-aware ai assistance to support learning and assessment. In *Extended Abstracts of the 2026 CHI Conference on Human Factors in Computing Systems*, page 5, Barcelona, Spain, Apr. 2026. ACM. Extended Abstracts, April 13–17, 2026, in press.

[9] O. Joshi, K. Mundra, D. Yadav, and S. Das. Ai coding assistants in competitive programming: Empirical studies from india on human-ai interactions in learning, problem solving, and curiosity development perspectives. In *International Conference on Artificial Intelligence in Education*, pages 181–194. Springer, 2025.

[10] W. Matcha, D. Gašević, N. A. Uzir, J. Jovanović, and A. Pardo. Analytics of learning strategies: Associations with academic performance and feedback. In *Proceedings of the 9th international conference on learning analytics & knowledge*, pages 461–470, 2019.

[11] S. Peng, E. Kalliamvakou, P. Cihon, and M. Demirer. The impact of ai on developer productivity: Evidence from github copilot. *arXiv preprint arXiv:2302.06590*, 2023.

[12] C. Piech, J. Huang, A. Nguyen, M. Phulsuksombati, M. Sahami, and L. Guibas. Learning program embeddings to propagate feedback on student code. In *International conference on machine Learning*, pages 1093–1102. PMLR, 2015.

[13] T. Sinha, N. Li, P. Jermann, and P. Dillenbourg. Capturing "attrition intensifying" structural traits from didactic interaction sequences of mooc learners. In *Proceedings of the EMNLP 2014 workshop on analysis of large scale social interaction in MOOCs*, pages 42–49, 2014.

[14] P. Vaithilingam, T. Zhang, and E. L. Glassman. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In *Chi conference on human factors in computing systems extended abstracts*, pages 1–7, 2022.

[15] P. H. Winne and R. S. Baker. The potentials of educational data mining for researching metacognition, motivation and self-regulated learning. *Journal of Educational Data Mining*, 5(1):1–8, 2013.

[16] B. J. Zimmerman. Becoming a self-regulated learner: An overview. *Theory into practice*, 41(2):64–70, 2002.