[Short]From Decoding to Encoding: Cognitive Reorganization of Human Coding under Generative AI

Jongwon Ryu, Junyeong Kim
Department of Artificial Intelligence
Chung-Ang University
Seoul, 06974
[fbwhddnjs511, junyeongkim]@cau.ac.kr

Abstract

The emergence of large generative models has reshaped human coding cognition. Traditional programming requires both encoding-conceptualizing a problem, and decoding-translating concepts into code. Generative models automate much of the decoding process, externalizing expressive cognition and allowing humans to focus on higher-level conceptual reasoning. This study examines how such automation reorganizes cognitive processes in programming. In a within subject pilot experiment , participants completed equivalent coding tasks with and without assistance from ChatGPT-5. Behavioral, verbal, and self-report data revealed greater conceptual focus and reduced low-level control effort under AI-assisted conditions. Overall, the findings suggest that generative AI redistributes cognitive load, shifting programming from an implementation task to a conceptual design activity.

1 Introduction

The rise of large generative models has fundamentally changed how humans engage in programming. Traditional coding involves two distinct cognitive stages: *encoding*-conceptualizing and structuring a problem, and *decoding*-translating concepts into executable code. These stages rely on separable cognitive subsystems, with decoding often demanding greater cognitive precision and working-memory resources [2, 5, 9, 3].

Generative models now automate much of the decoding process [4, 1]. By offloading expressive implementation to an external agent, programmers are relieved from micro-level control and can redirect cognitive resources toward higher-level conceptual reasoning [6, 8]. This shift reframes programming from "how to implement" to "what to implement," signaling a fundamental reorganization of human coding cognition [7].

Theoretical perspective. We formalize this transformation through an *encoding–decoding cognitive model*, where total cognitive load is distributed between conceptual (encoding) and expressive (decoding) processes. When a generative model assumes decoding, human cognitive resources are reallocated toward encoding, producing a functional segregation that explains observed performance gains not as simple efficiency, but as a reorganization of cognitive structure [9, 3].

Research objective. While prior studies on AI-assisted programming have reported productivity improvements, they seldom explain the underlying cognitive mechanisms. This work addresses that gap by examining how generative models externalize expressive cognition and redistribute mental effort toward conceptual reasoning [6, 1].

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

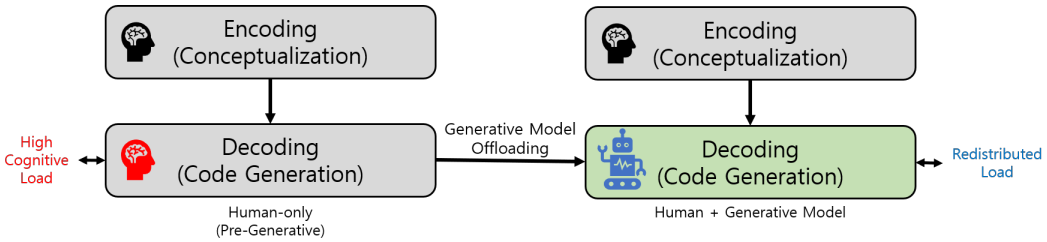

Figure 1: Overview of the proposed encoding–decoding cognitive framework.

2 Method and Results

We conducted a within-subject pilot study (n=5) to examine how generative models redistribute cognitive roles in programming. Each participant solved two short Python tasks under **Control** (no assistance) and **Generative** (ChatGPT-5 assistance) conditions, counterbalanced to mitigate order effects [9, 3]. Tasks emphasized reasoning over syntax and were designed to be solvable within 30 lines of code [4]. Participants verbalized their thought process during each session [5], and behavioral, verbal, and self-report data were collected.

Quantitative analysis compared *planning ratio*, *task performance*, and *cognitive ratings*. As shown in Table 1, generative assistance increased pre-coding planning time (Δ**=+0.083**) and conceptual focus ($\Delta Q2$**=+1.5**), while reducing micro-level control effort ($\Delta Q1$**=-1.2**) and total completion time (Δ**=-93.2s**). Verbal analysis revealed that participants produced more conceptual and fewer syntactic utterances, indicating a cognitive shift from expressive to conceptual reasoning [2, 6].

Table 1: Paired comparison of key metrics between conditions (T = Generative, C = Control, $n = 5$).

Metric	**Mean Δ(T-C)**	**t**	**p**	**Cohen's d**
Planning ratio	+0.083	5.21	0.007	2.3
First pass time (s)	-93.2	-3.88	0.017	1.7
Tests passed	+0.4	1.94	0.12	0.9
Q1 Micro control burden	-1.2	-4.12	0.014	1.8
Q2 Conceptual focus	+1.5	4.37	0.011	1.9
Q3 Overall demand	-0.7	-2.26	0.086	1.0

3 Discussion and Conclusion

Results suggest that generative models do more than improve efficiency-they restructure cognitive processes in programming. By externalizing the decoding process, humans allocate more cognitive resources to conceptual reasoning, supporting a *functional segregation* between human and AI cognition [9, 6]. Qualitative analysis of verbal protocols further revealed a linguistic shift: participants in the generative condition framed their intentions conceptually (e.g., "I need a structure for fast lookup") rather than procedurally (e.g., "write a loop to sort"), illustrating how expressive cognition was externalized to the model. This implies that generative AI functions as a *cognitive redistribution mechanism*, transforming programming from implementation to conceptual design.

Although limited by sample size (n=5), consistent behavioral, subjective, and linguistic patterns validate this hypothesis. Future work should expand to larger populations and incorporate multimodal cognitive measures (e.g., eye-tracking, EEG) to examine real-time cognitive load redistribution. Overall, these findings move beyond the "AI for productivity" narrative toward a view of AI as a catalyst for cognitive transformation.

References

[1] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N Bennett, Kori Inkpen, et al. Guidelines for human-ai interaction. In *Proceedings of the 2019 chi conference on human factors in computing systems*, pages 1–13, 2019.

[2] John R Anderson. *The architecture of cognition*. Psychology Press, 2013.

[3] Alan Baddeley. Working memory. *Memory*, pages 71–111, 2020.

[4] Christian Bird, Denae Ford, Thomas Zimmermann, Nicole Forsgren, Eirini Kalliamvakou, Travis Lowdermilk, and Idan Gazit. Taking flight with copilot: Early insights and opportunities of ai-powered pair-programming tools. *Queue*, 20(6):35–57, 2022.

[5] Allen Newell, Herbert Alexander Simon, et al. *Human problem solving*, volume 104. Prentice-hall Englewood Cliffs, NJ, 1972.

[6] Ben Shneiderman. Human-centered artificial intelligence: Three fresh ideas. *AIS Transactions on Human-Computer Interaction*, 12(3):109–124, 2020.

[7] Ben Shneiderman and Pattie Maes. Direct manipulation vs. interface agents. *interactions*, 4(6): 42–61, 1997.

[8] S Shyam Sundar. Rise of machine agency: A framework for studying the psychology of human–ai interaction (haii). *Journal of computer-mediated communication*, 25(1):74–88, 2020.

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

A Experimental Setup

A.1 A.1 Experimental Design

A within-subject design was adopted to examine cognitive role redistribution during programming with and without generative-model assistance. Each participant completed two programming tasks of comparable complexity under two counterbalanced conditions: (1) **Control**—coding without external assistance, and (2) **Generative**—coding with ChatGPT-5 assistance. This structure isolates the cognitive effects of model assistance while holding individual skill and task difficulty constant [9, 3, 5].

A.2 A.2 Participants

Five participants (n=5) with 1–5 years of Python programming experience were recruited. All reported regular exposure to IDE-based coding but no prior training in cognitive or experimental research, ensuring naturalistic programming behavior [2]. Each session lasted approximately 30 minutes.

A.3 A.3 Experimental Environment

All experiments were conducted on standard laptops (Python 3.10, VSCode IDE). The generative model used in the assisted condition was **ChatGPT-5** (April 2025 release), accessed through the official web interface with default settings. Internet access and external searches were disabled for both conditions.

A.4 A.4 Task Materials

Two programming tasks were designed to emphasize conceptual reasoning while minimizing boiler-plate code:

- **Task A: File Extension Counter.** Count occurrences of file extensions from a list of filenames, ignoring case and excluding files without extensions.
- **Task B: URL Query Parser.** Parse and decode query strings into key–value pairs, merging duplicate keys into lists.

Each task contained three predefined unit tests for scoring. Full prompts and test cases are provided in the supplementary materials [4].

A.5 A.5 Procedure

Participants first completed a 2-minute orientation and a short think-aloud practice session [5]. They then performed two 10-minute programming tasks (one per condition) with a 2-minute break in between. In the Generative condition, participants interacted freely with ChatGPT-5 using natural-language prompts and could copy or edit its responses [1, 7]. The Control condition disallowed any external assistance or web search. All sessions were screen-recorded, and verbalizations were transcribed for later analysis.

A.6 A.6 Data Collection

We collected three complementary data streams to capture both behavioral and cognitive dimensions of programming:

- **Verbal protocol:** Think-aloud recordings were transcribed into sentence-level utterances for qualitative analysis [5].
- **Behavioral logs:** Automatic timestamps captured first keypress, first successful test, and total task time.
- **Self-report:** Three 7-point Likert items measured (Q1) perceived micro-level control burden, (Q2) conceptual focus, and (Q3) overall mental demand [9, 3].

For the Generative condition, all ChatGPT-5 prompts and responses were also logged for post-hoc linguistic classification (conceptual vs. imperative prompts) [8].

A.7 A.7 Measurement Definitions

- **Planning ratio:** $(t_{\text{first\_keypress}} - t_{\text{start}})/t_{\text{total}}$.
- **First pass time:** Elapsed time until the first successful test case.
- **Tests passed:** Number of passing unit tests (0–3).
- **Q1–Q3:** 7-point Likert responses on perceived cognitive effort and focus.

A.8 A.8 Post-task Questionnaire

After each task, participants rated the following items on a 7-point Likert scale (1 = strongly disagree, 7 = strongly agree):

- **Q1.** "I had to manage many low-level implementation details." (micro-level control burden)
- **Q2.** "I focused mainly on understanding and structuring the problem conceptually." (conceptual focus)
- **Q3.** "The task required a high level of mental effort." (overall mental demand)

After both conditions, participants also answered an open-ended reflection question: *"What felt most different between the two coding experiences?"*

A.9 A.9 Analysis

Quantitative data were analyzed using paired *t*-tests with Cohen's *d* for effect sizes. Given the small sample size, results are treated as exploratory indicators rather than confirmatory evidence. Qualitative analysis focused on the proportion of encoding-related utterances and the frequency of encoding–decoding transitions, consistent with cognitive process-tracing methods [2, 6].

A.10 A.10 Analysis Tools

All data were processed in Python using `pandas`, `SciPy`, and `matplotlib`. Raw behavioral logs, anonymized transcripts, and analysis scripts will be made publicly available upon publication.