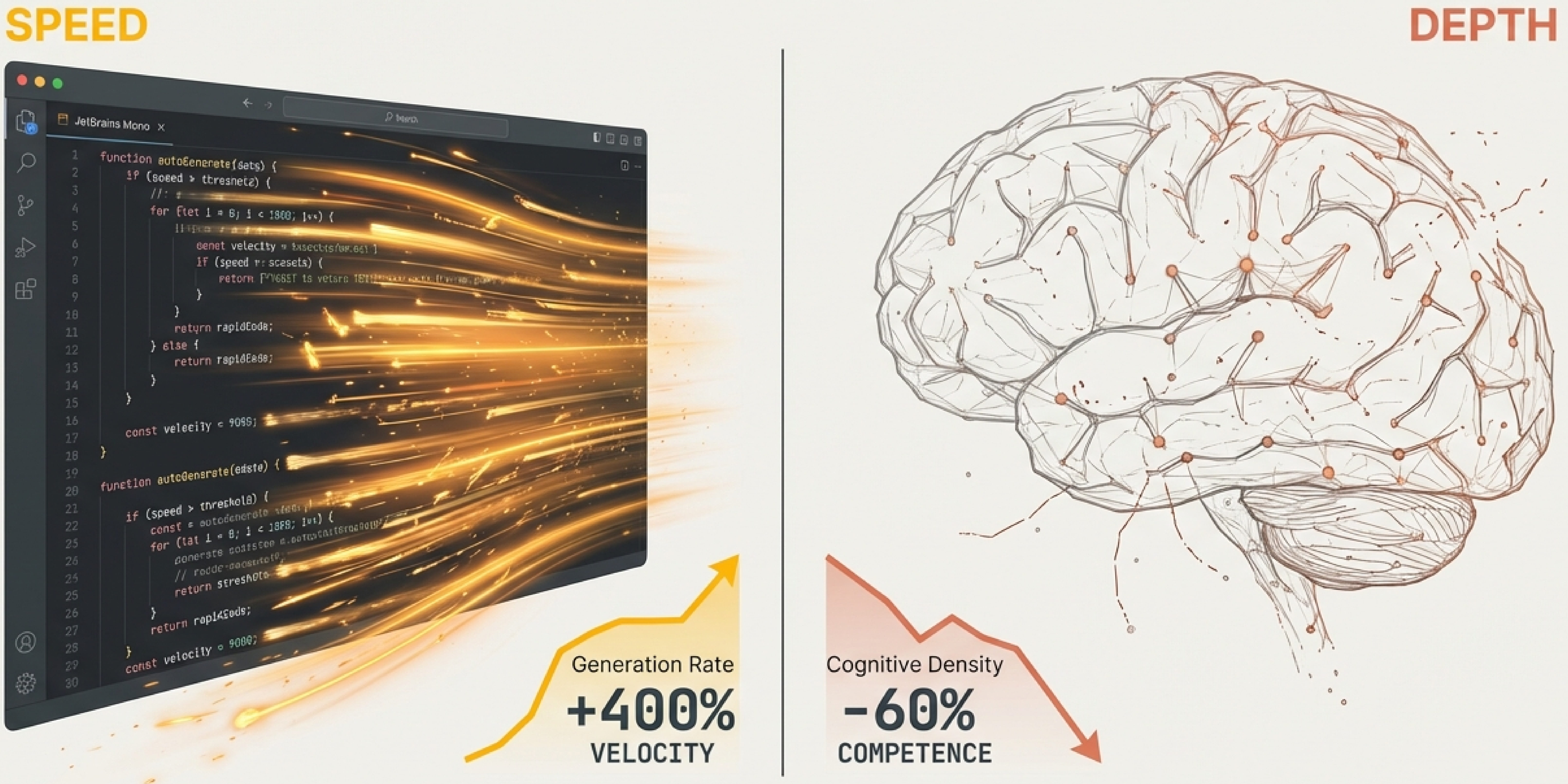
The Skill Formation Paradox

A computational analysis of how AI coding tools boost short-term speed while eroding long-term competence.



Based on the KDD '26 simulation model of novice developer skill acquisition.

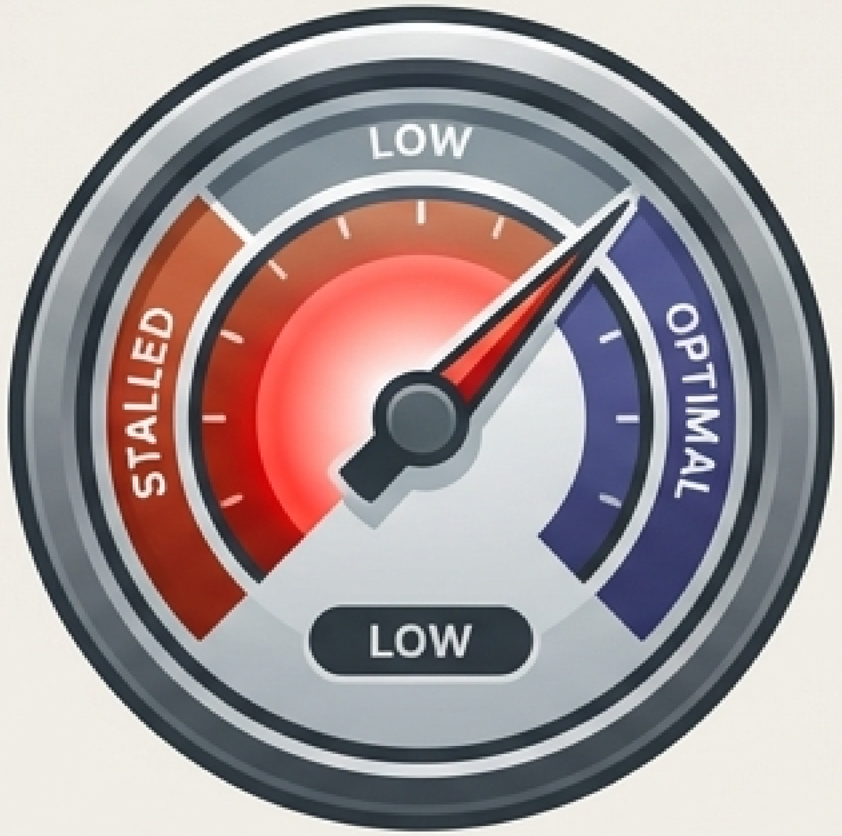
Executive Summary: The hidden cost of frictionless coding

The Trap

Unrestricted AI usage creates a Productivity-Skill Dissociation. Novices appear 55% faster but suffer massive skill attrition.

Cohen's *d* = -1.04

The Mechanism

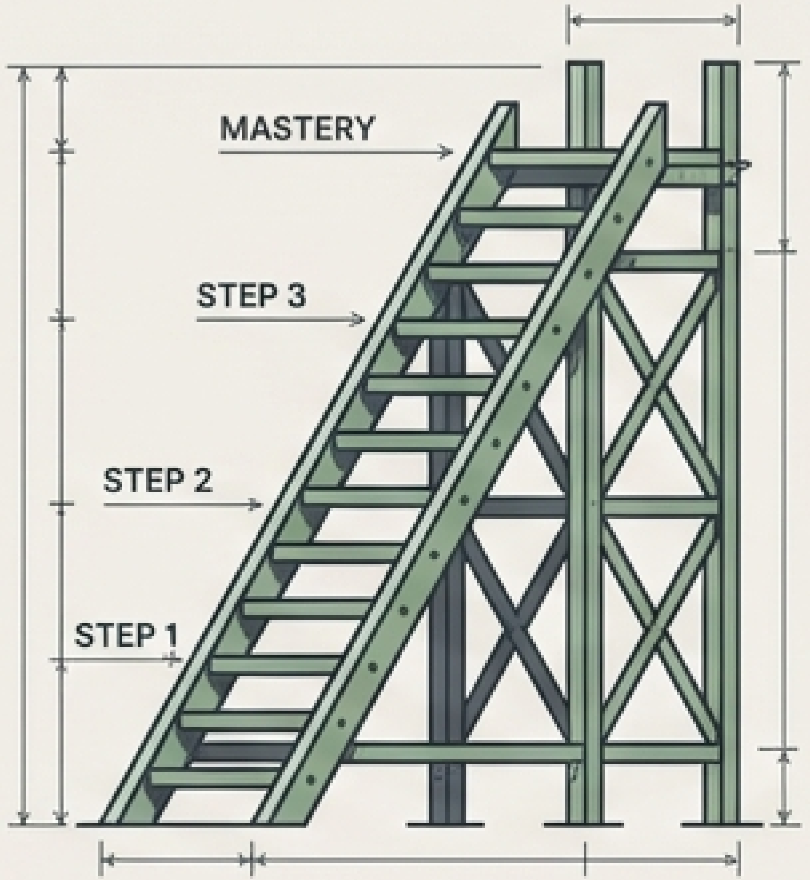


Learning requires Processing Depth. When AI lowers cognitive engagement below a critical threshold, learning stalls.

Threshold: 0.75

Cognitive engagement 0.75

The Solution



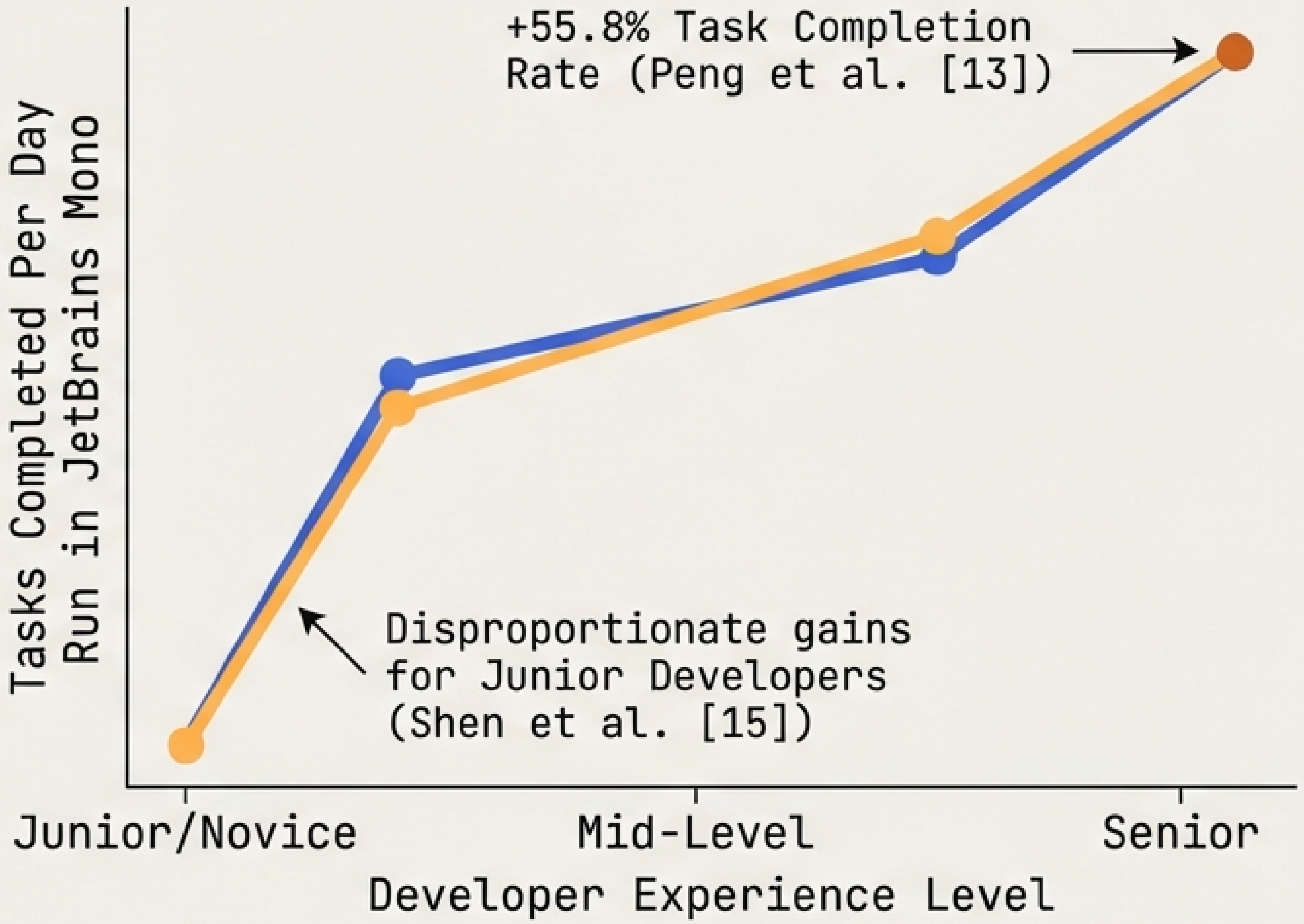
We do not need to ban AI. Implementing 'Scaffolded Engagement' (explain-before-accept) preserves nearly 100% of skill growth while maintaining utility.

Skill Impact:

***d* = -0.04 (Negligible)**

The current consensus is that AI is a productivity engine.

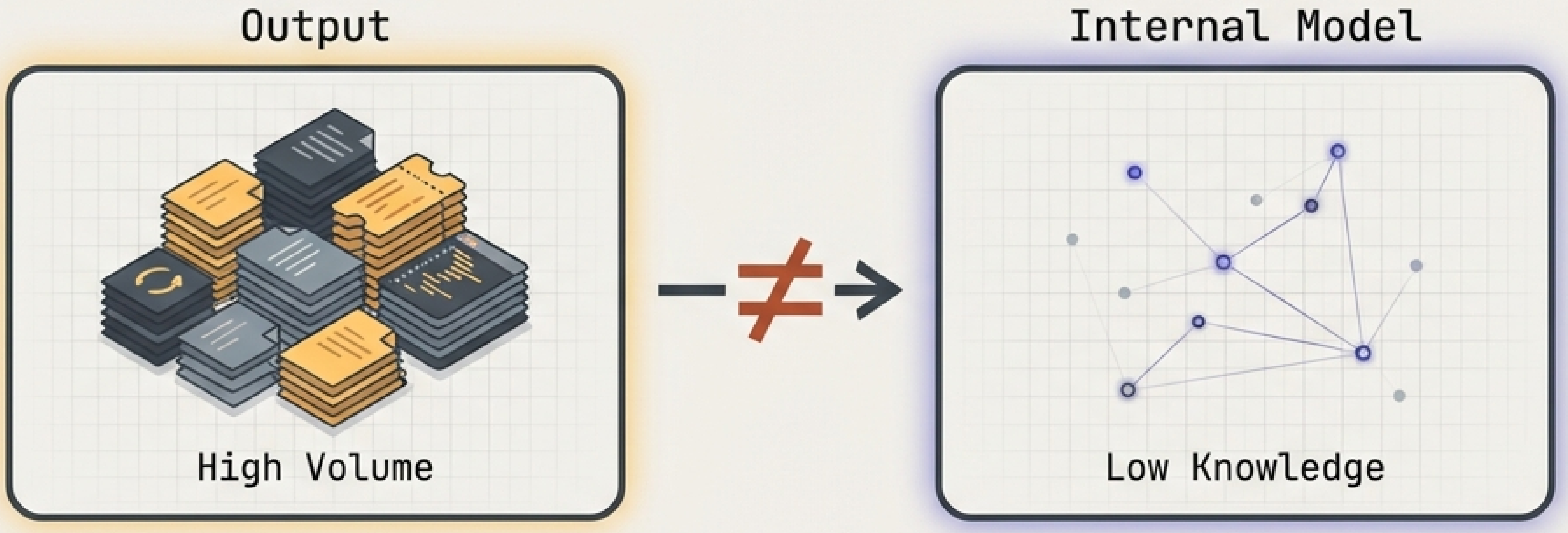
Tools like GitHub Copilot and Claude have transformed workflows. For a novice, the friction of syntax and boilerplate has vanished. Recent empirical studies validate this feeling of speed.



But what is replacing the friction?

Completing a task is not the same as learning how to do it.

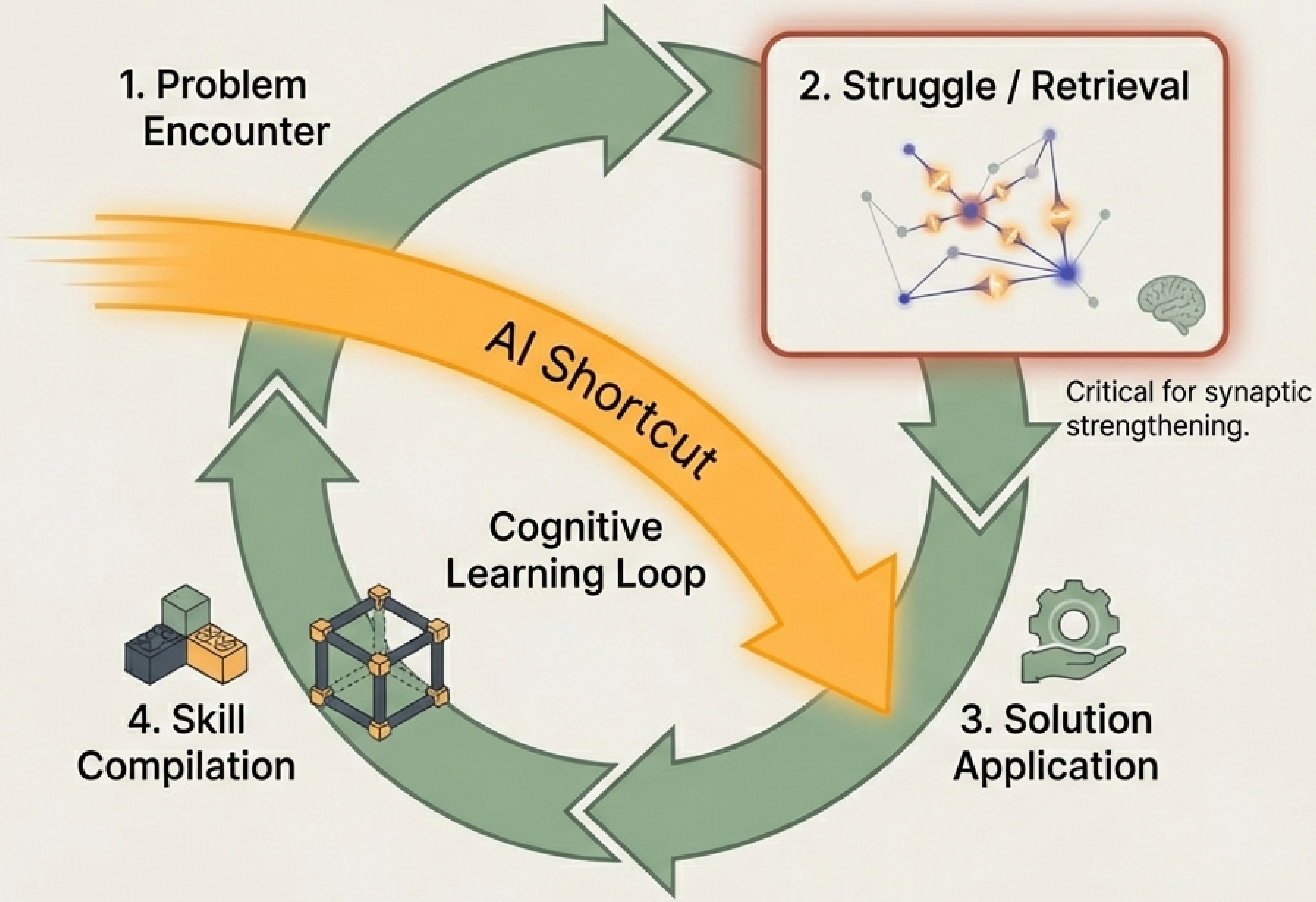
The Productivity–Skill Dissociation



A novice completing tasks rapidly with AI is not necessarily compiling the knowledge required to repeat the feat. If we prioritize output over understanding, we create a cohort of “hollow” developers—productive only when the AI is active, and helpless when it is removed.

“Unrestricted AI users look smarter but are actually learning less.”

Learning requires struggle, retrieval, and "Desirable Difficulty".



Retrieval-Based Strengthening: Skills consolidate when we struggle to recall them.

Desirable Difficulty: Moderate challenge prevents the learning curve from flattening to zero.

The Investigation: We simulated 240 novice developers over a virtual year.

Virtual Laboratory

The Subjects

240 Novice Developers

(0-2 years experience, N ~ 0.20 initial skill)

The Timeline

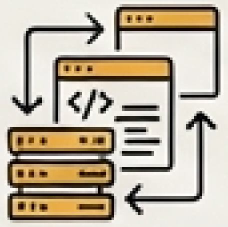
252 Working Days

(12 Months of simulated cognitive evolution)

The Workload

5 Coding Tasks / Day

Varying difficulty & skill activation

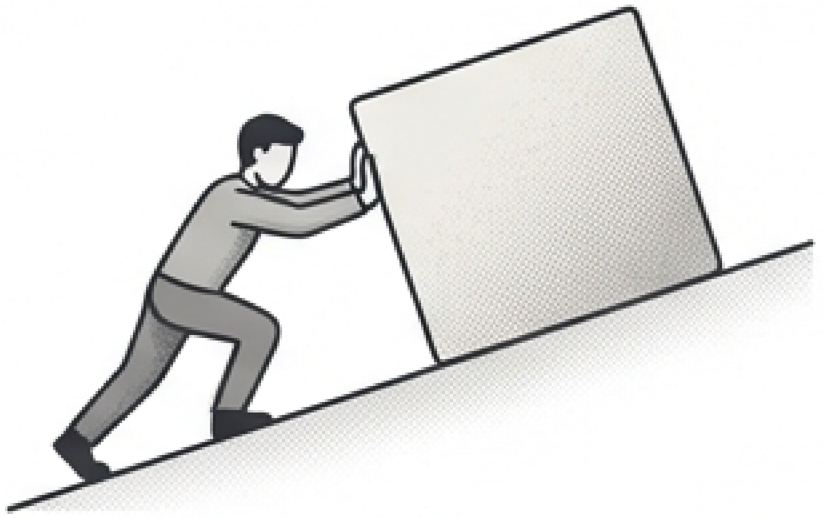


The Goal

To observe skill evolution when tools are REMOVED.

Three distinct modes of AI engagement were tested.

Control (No AI)

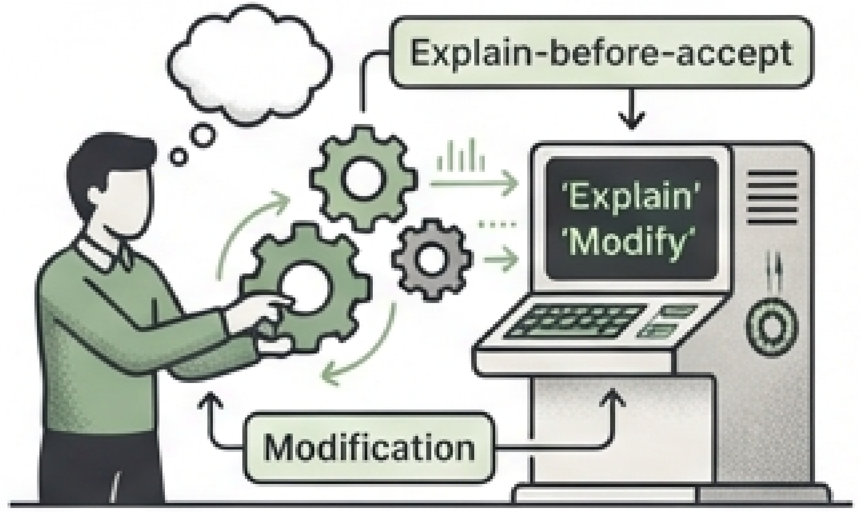


Hard work. High friction.
Developers must solve problems independently.

Unrestricted AI

Full access. Passive acceptance.
The AI provides code, and the developer accepts it with minimal review (The “Copy/Paste” standard).

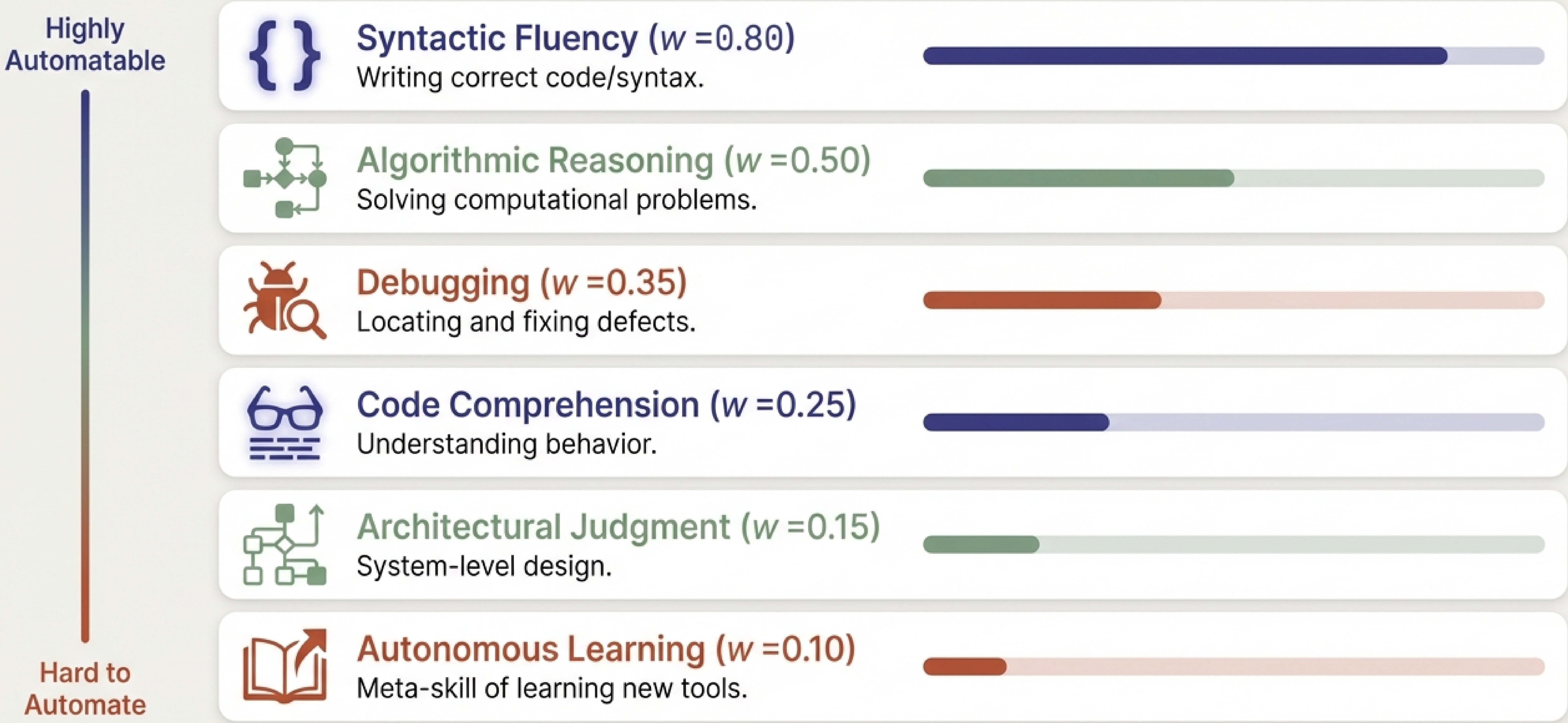
Scaffolded AI



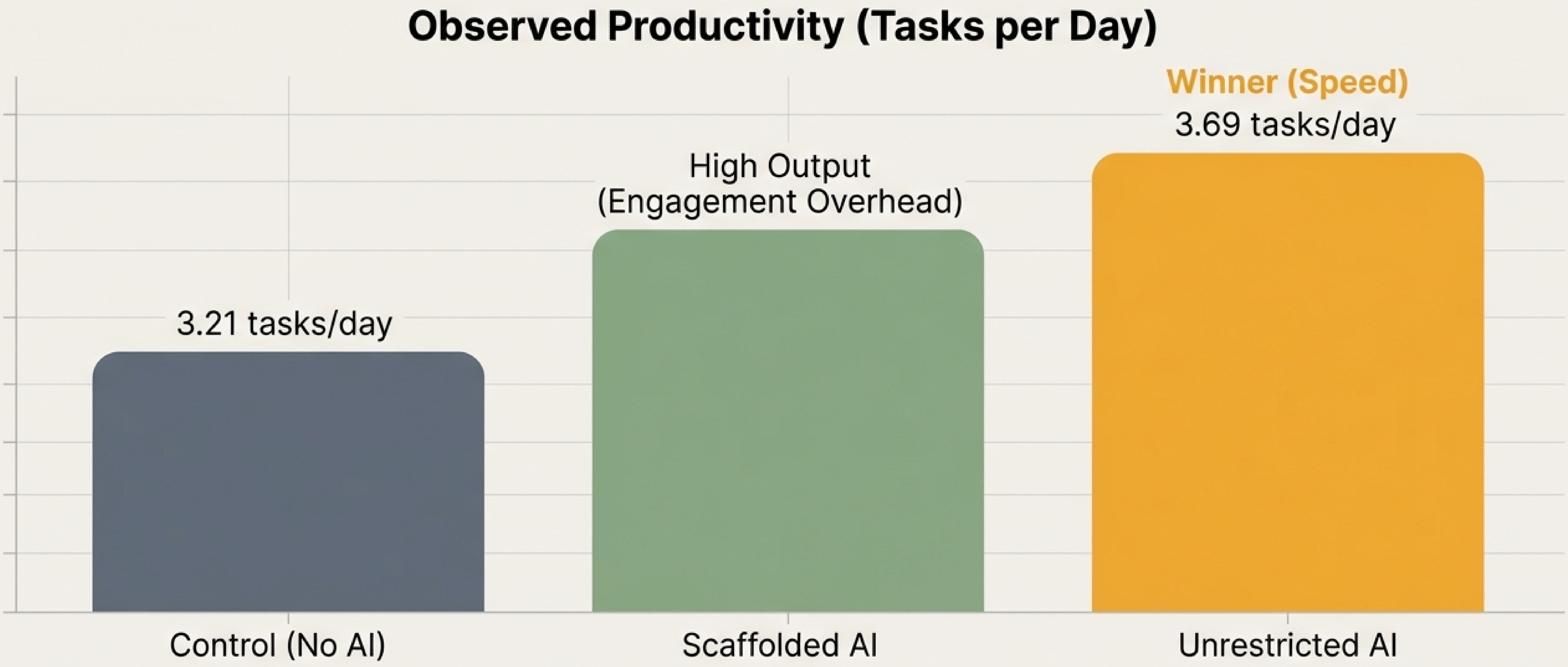
Mandatory engagement.
AI access is granted, but requires “Explain-before-accept” and modification before proceeding.

Competence was measured across six distinct cognitive dimensions.

Ranked by AI Automation Weight (w)



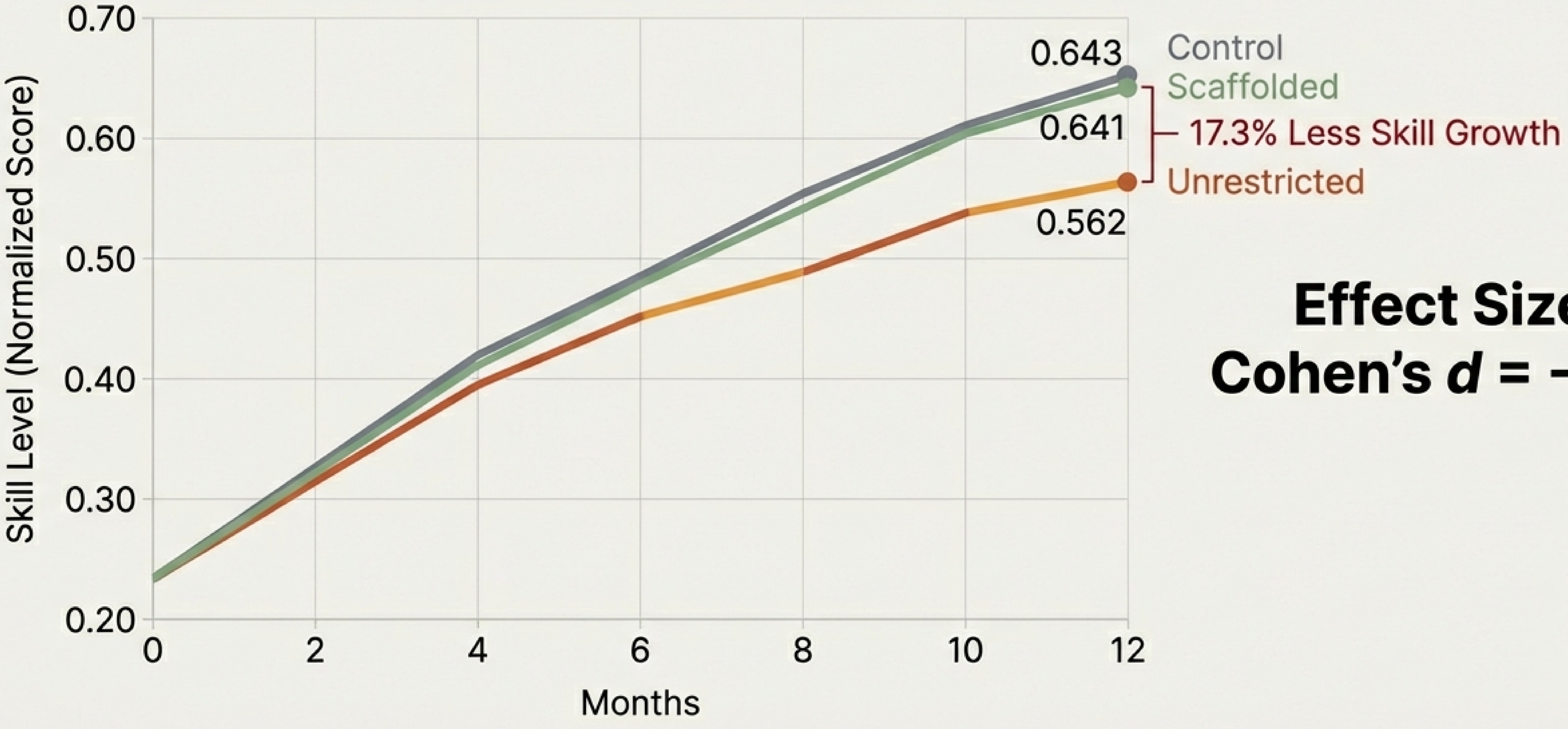
Unrestricted AI creates the most productive employees...



If you measure success by tickets closed or commits pushed, Unrestricted AI is the obvious choice.

...but it produces the least competent engineers.

Underlying Skill Level (Tool-Removed Assessment)



Effect Size:
Cohen's *d* = -1.04

The skills we delegate to AI are the skills that atrophy.

Detailed Skill Dimension Analysis (Normalized Scores)

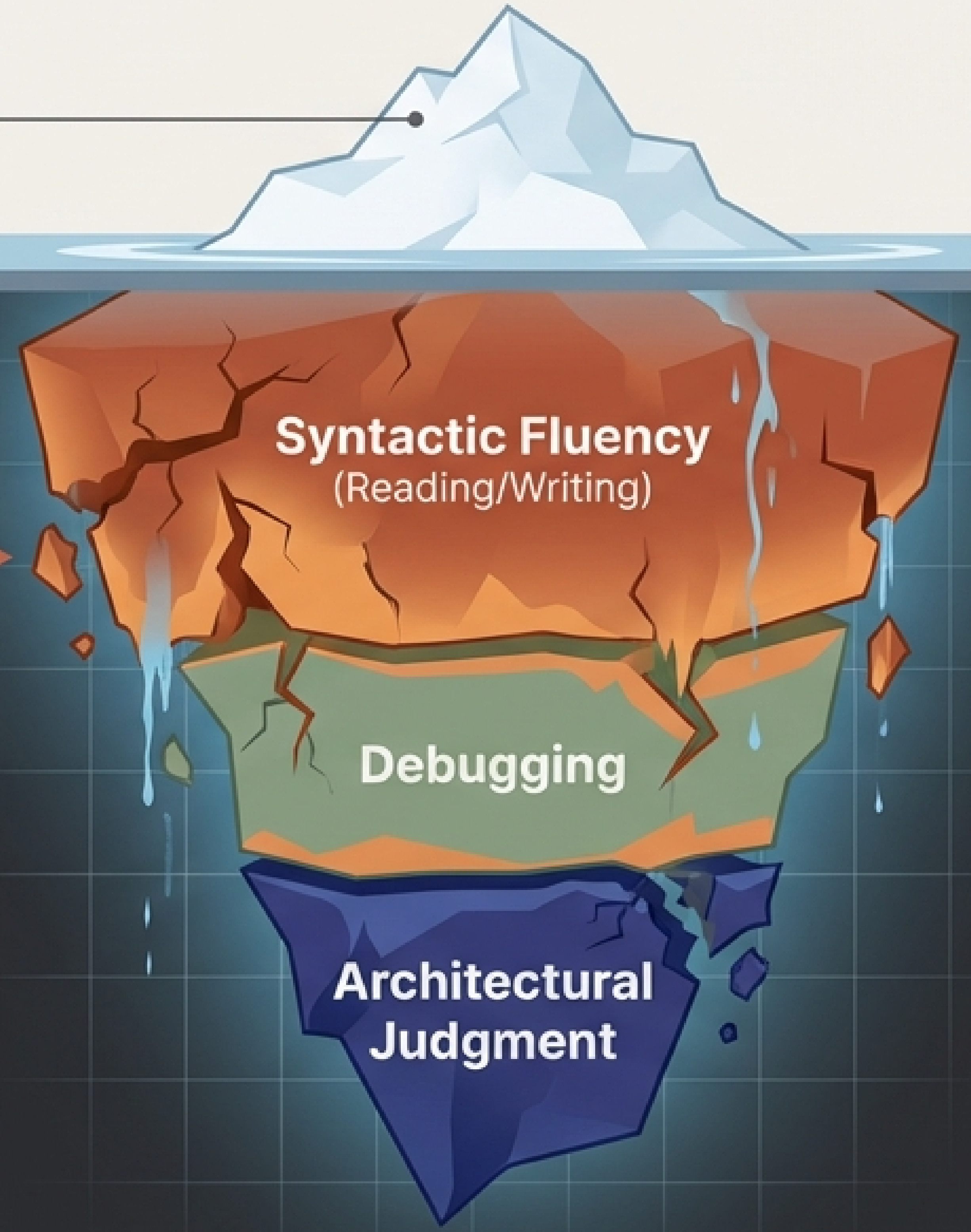
	Syntactic Fluency	Semantic Understanding	Logic & Reasoning
Unrestricted AI	0.390	0.517	0.623
Scaffolded AI	0.483		0.864
Control (No AI)	0.620	0.748	0.736
	Syntactic Fluency	Semantic Understanding	Logic & Reasoning

Deep Skill Loss (0.390 vs 0.651 Control)
Effect Size: d = -5.10

The correlation is clear (p = -0.94). The more the AI helps (w), the less the human learns. We are trading fundamental fluency for speed.

Why the loss of “Syntactic Fluency” matters.

Observable Output
(Code generation)
JetBrains Mono

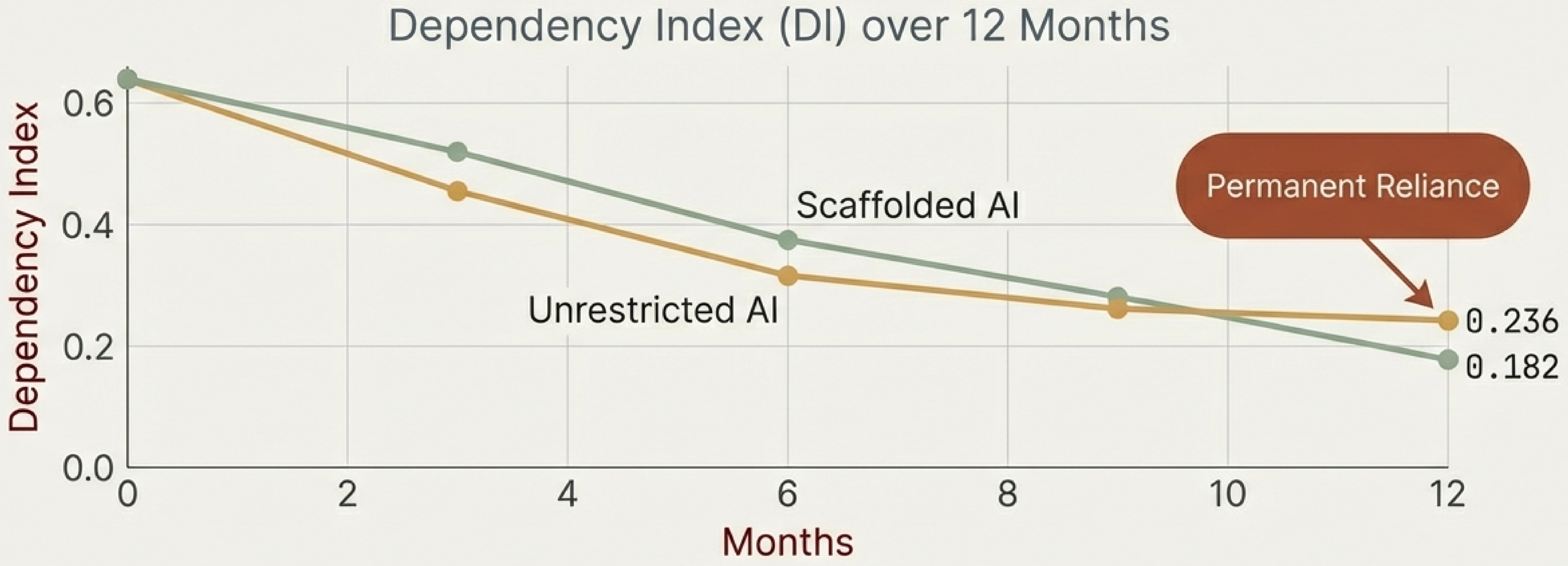


Burnt Terracotta
If this foundation melts, the ability to Debug and Architect collapses.

A developer with low syntactic fluency (d = -5.10) becomes a “Black Box Operator.” They can generate code, but lack the granularity of understanding to fix it when it breaks.

Crimson Pro with
JetBrains Mono data

The Dependency Trap: Unrestricted users never wean off the tool.



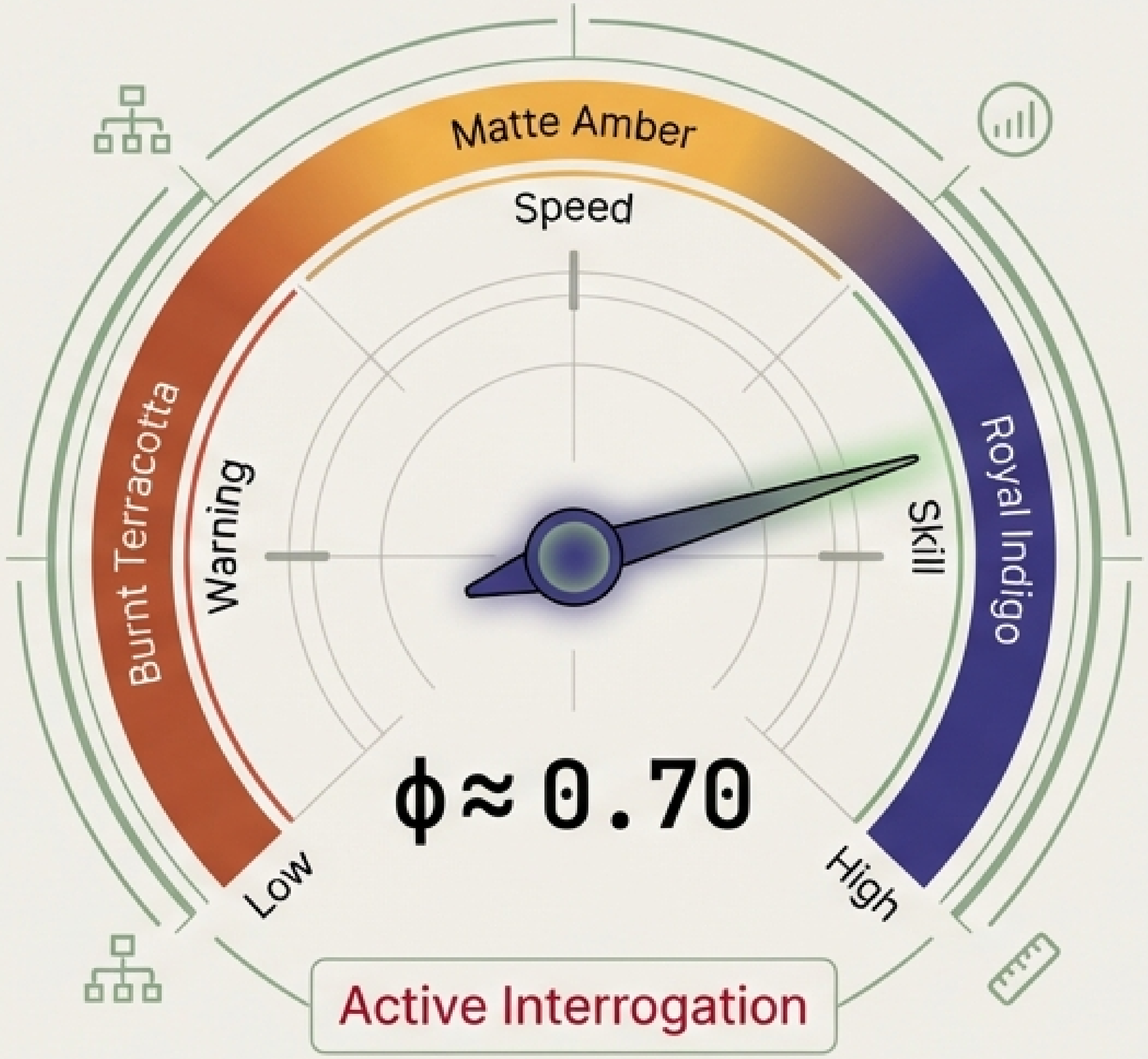
After one year of practice, Unrestricted users are still heavily reliant on AI to perform basic tasks.

The mechanism of failure is low “Processing Depth”.

Unrestricted AI

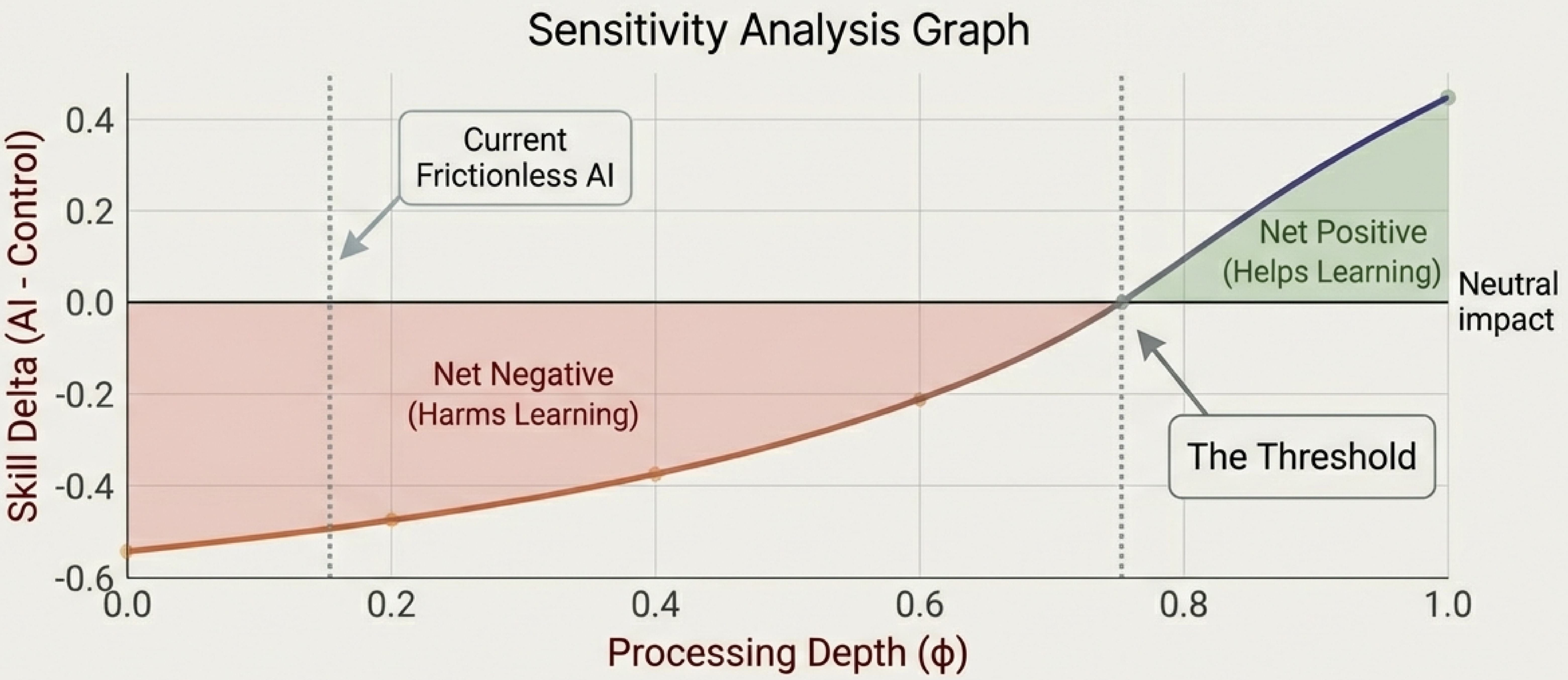
Tab to complete.
Low depth = Low learning signal.

Scaffolded AI



Read, Verify, Edit.
High depth = Strong learning signal.

The Crossover Threshold is 0.75.

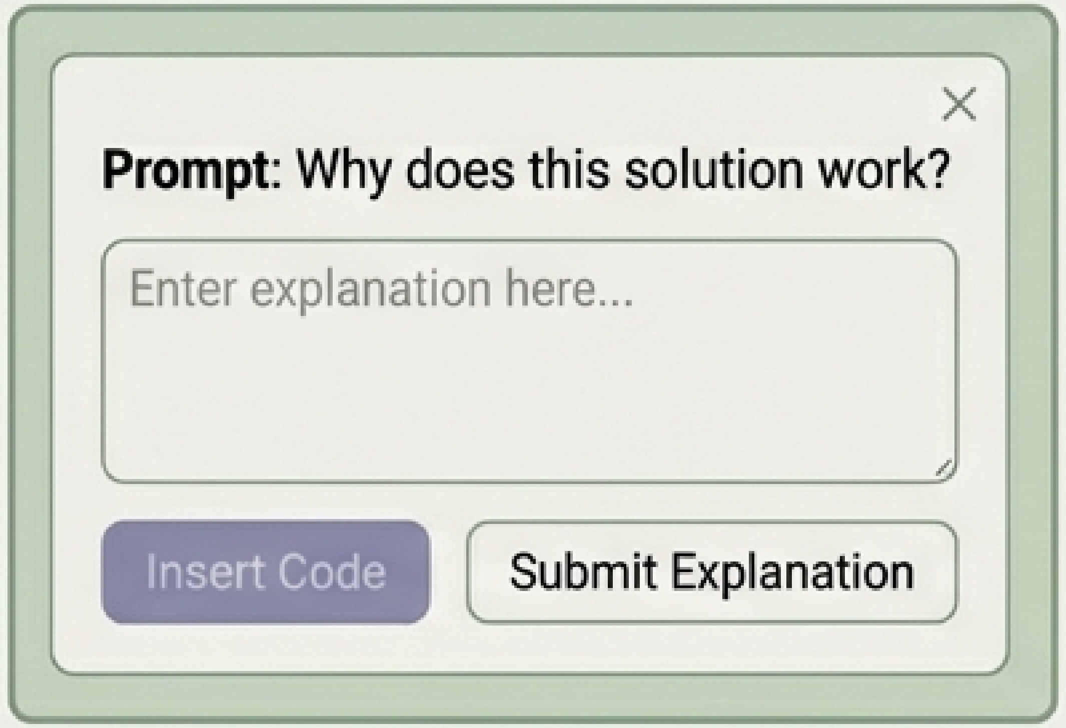


To transition from harmful dependency to beneficial learning, AI systems must actively increase processing depth to meet or exceed the 0.75 threshold.

The Solution: “Scaffolded Engagement”.

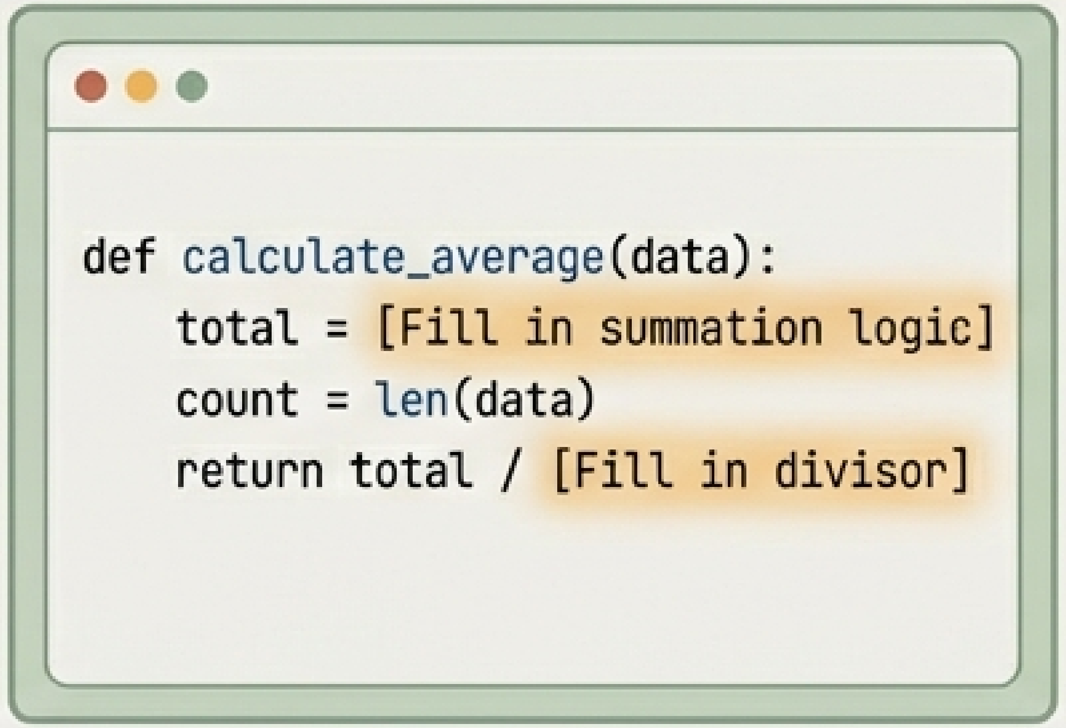
Reintroducing specific friction to ensure cognitive processing.

1. Explain-before-accept



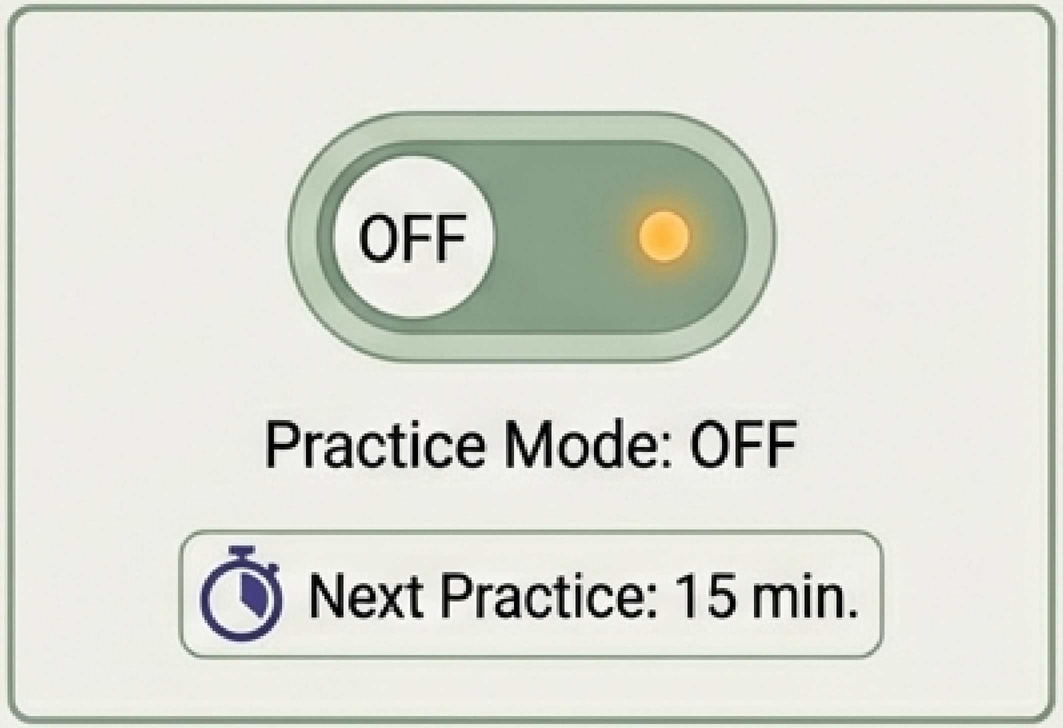
Visual: A dialog box UI pop-up asking “Why does this solution work?” before code can be inserted.

2. Modification Prompts



Visual: AI suggesting a template rather than finished code, requiring the user to fill in gaps.

3. Interleaved Practice



Visual: A toggle switch turned “OFF” periodically.

Think of this as training wheels that force you to pedal, rather than an electric motor that does the work for you.

Scaffolding preserves learning without sacrificing utility.

Skill Growth vs Control

d = -0.04 ↓

Statistically indistinguishable from the Control group. No learning loss.

Algorithmic Reasoning Bonus

↑ **d = +0.34**

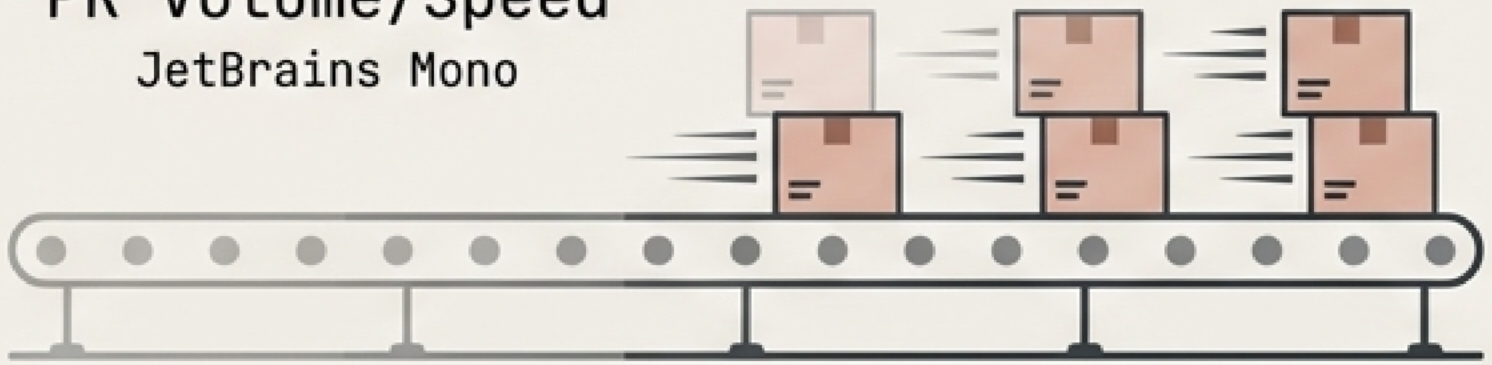
Scaffolded users actually outperformed Control in complex reasoning.

Active engagement filters the AI's speed through the human's cognitive process, allowing compilation of knowledge to occur.

For Engineering Managers: Trust, but verify (without the tool)

The Risk

Evaluating juniors solely on pull request volume or speed will hide competency gaps.



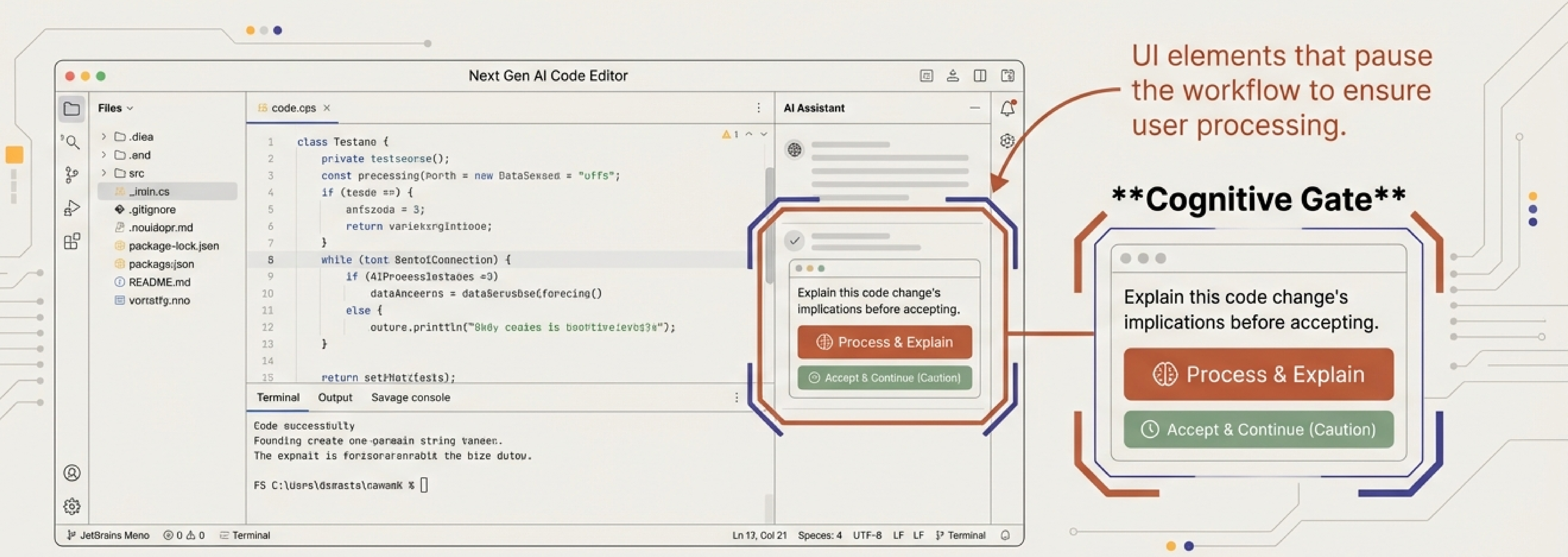
You are measuring the tool, not the talent.

The Fix

Tool-Removed Assessments
Implement periodic whiteboarding or unassisted debugging sessions to sessions to check for hidden dependencies.

Value ‘Why’ over ‘What’
Code reviews must focus on the developer's ability to explain the logic, not just the correctness of the output.

For Tool Designers: Friction is a feature, not a bug.



- Stop optimizing for "one-click" acceptance.
- Implement Progressive Withdrawal: Like training wheels, AI assistance should decrease as user skill increases (The "Expertise Reversal Effect").

Speed is not Skill.

Unrestricted AI creates a “Productivity-Skill Dissociation,” boosting output while hollowing out capability. We must design AI interaction to **augment** human cognition, not to replace the struggle that creates it.