Comprehension debt & cognitive surrender

The hidden costs of AI assisted coding

Code generation has become cheap

Evaluation did not

Volume. An agent produces more code in a minute than you can read in ten.

Inversion. A junior can now generate faster than a senior can critically audit. The rate limit that used to make review meaningful is gone.

So review changed jobs. It used to be a quality gate that forced comprehension. It is now a throughput problem.

Emergent concepts from AI assisted coding

Surrender is how you take on the debt. Comprehension debt is the bill

Comprehension debt ⚠️

The gap between how much code exists in your system and how much of it any human genuinely understands.

- The AI’s output becomes our output
- We stop constructing the answers entirely

Cognitive offloading ✅

You hand off the how and keep the what. You still judge whether the result is sensible, and you intervene when it isn’t.

- Using a calculator
- Relying on GPS

Cognitive surrender ⚠️

The moment you stop constructing an answer and adopt the AI's instead so there's no view of your own left to compare it against.

- Cognitive surrender is the means to increase the comprehension debt in a project

Why it happens so easily?

Everything that used to mean “understood” is now free

1. **It compiles.**
2. **It passes the linter.**
3. **The tests are green.**
4. **It reads like the rest of the file.**

CI checks	Type check	Test	Lint	Build
11ms	3s	2s	13s	3s
11ms	3s	2s	13s	3s

Every one of those signals used to be evidence a human had thought about it. Now they're free. Surface correctness is not systemic correctness.

Borrowed confidence

Invisible on the metrics

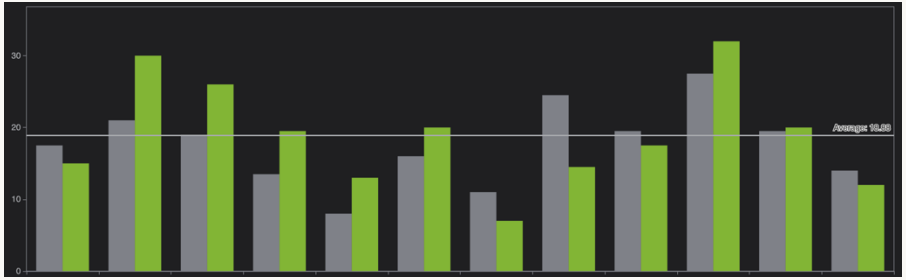
The psychology

- Wharton study (Shaw & Nave): 3 experiments, 1,372 participants
- Just *having* AI available was enough to trigger surrender
- When AI was wrong → accepted anyway **73%** of the time
- Confidence went **up** with AI present
  - Even on trials with wrong answers

The measurement gap

- Velocity ↑ · PR count ↑ · Coverage ✓
- None of these detect a comprehension deficit because it isn't measured or possible or measure anywhere

Velocity metrics might look immaculate



What’s the big deal?

Technical Debt vs. Comprehension Debt

Technical Debt

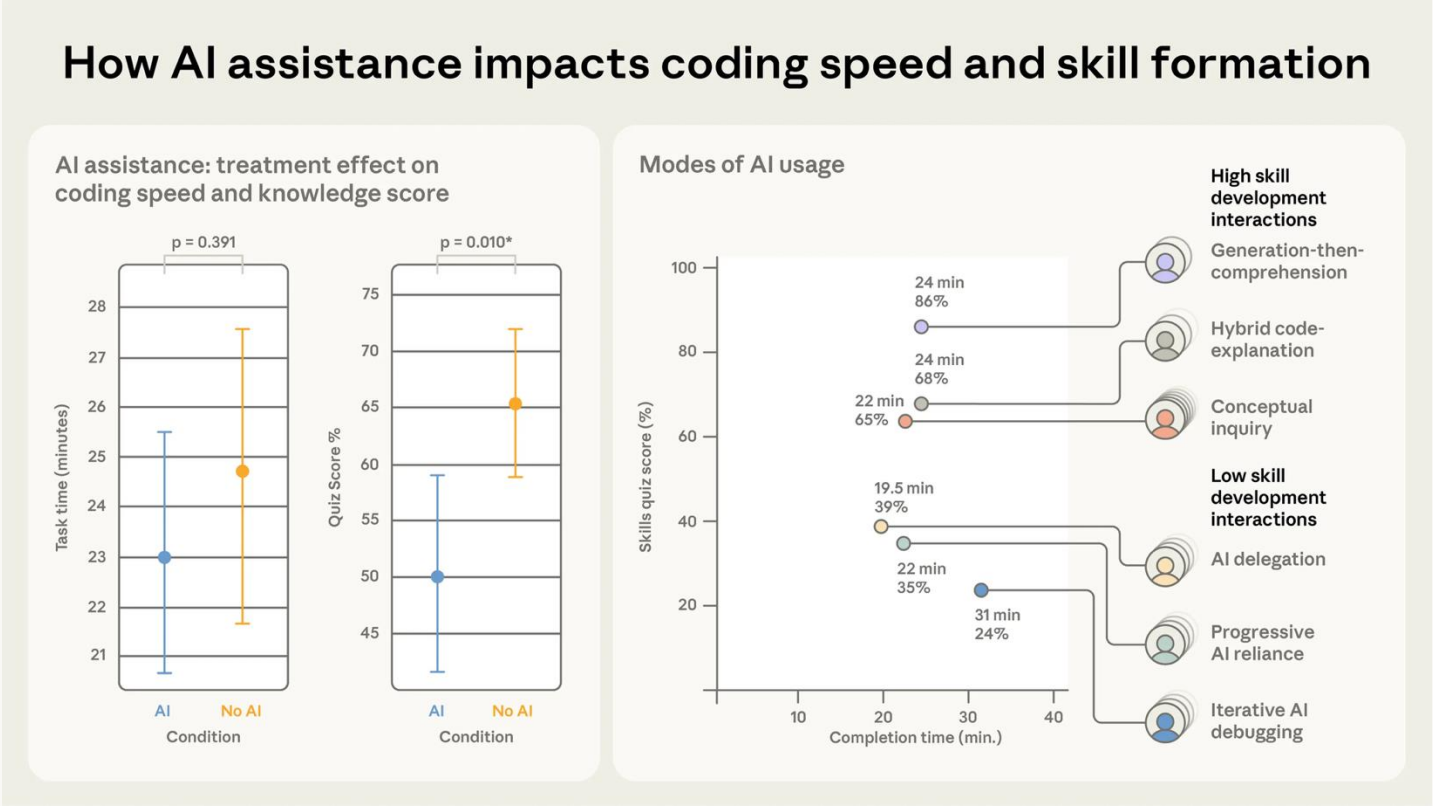
- A conscious tradeoff! You chose the shortcut
- Announces itself: friction, slow builds, tangled dependencies
- You know where it lives
- Can be scheduled and paid down

Comprehension Debt

- Accumulates without anyone deciding to take it on
- Invisible
  - codebase looks clean, tests stay green
- No one knows where it lives until something breaks
- Revealed suddenly, usually mid-incident

(Source: https://www.riotgames.com/en/news/taxonomy-tech-debt)

What the evidence actually says?



(Source: https://www.anthropic.com/research/AI-assistance-coding-skills)

This idea is forty years old

Programming as Theory Building

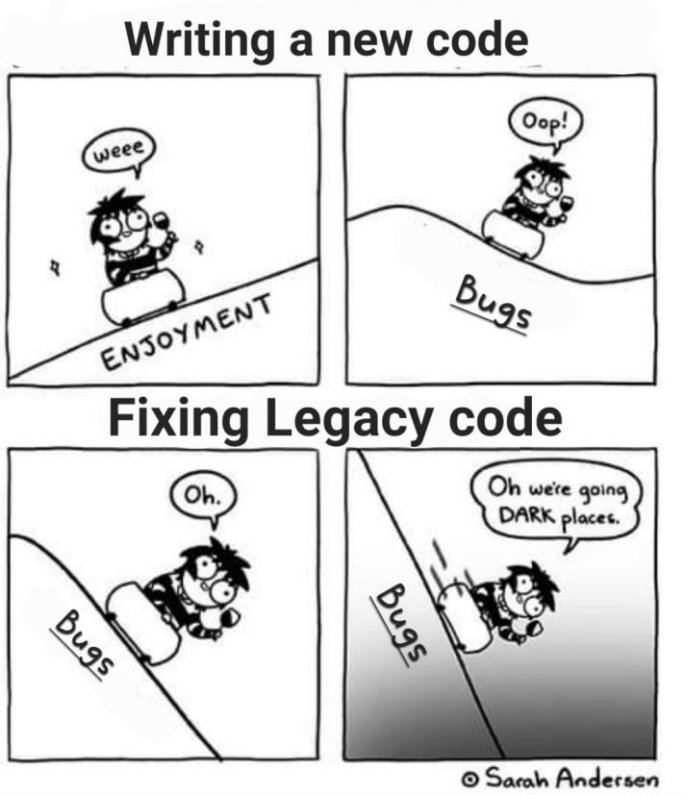
- The real product of programming isn't the code. It's a **theory**!
- A shared mental model of *how* the system works and *why* it's built that way
- Code is a byproduct of that theory which is an incomplete projection of it

What legacy code actually is:

- Not "old code" but code whose theory has decayed while the artifact survives
- The team is gone, the reasoning is gone, only the code remains

The connection to AI-generated code (large text, own line):

- When we merge code nobody has built a theory of, we don't create future legacy code rather we arrive at legacy code on day one.



(Author: https://x.com/SarahCAndersen)

The part that isn't just an engineering problem

It’s a promise we're making to paying customers

What I'm not claiming:

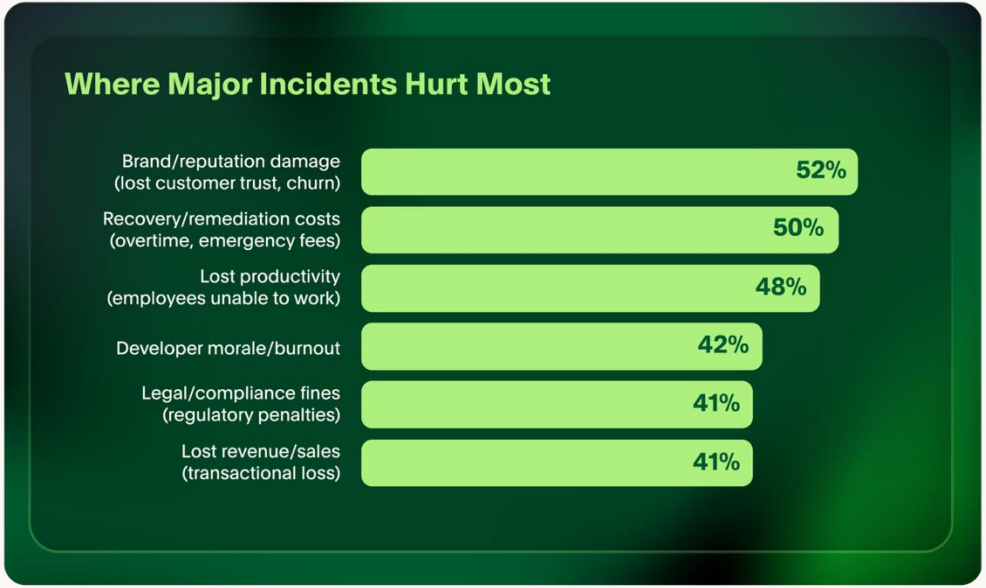
- No study shows AI-generated code causes more outages
- DORA links instability to change *volume* and missing controls not to defective code

The narrow claim that does hold:

- Recovery time during an incident is a function of comprehension nothing else
- Someone has to reconstruct *why* the system does what it does, under pressure
- Comprehension is exactly what we've been trading away

The cost, in numbers:

- 68% of orgs lose $300K+ per hour during an incident (PagerDuty, 2026)
- Unmanaged: small changes trigger failures → fixes cause regressions → cleanup *increases* risk



(Source: https://www.pagerduty.com/newsroom/2026-state-of-ai-first-operations/)

What we can do from next day

Habits we can adopt as individuals

- **Form an expectation before reading the output**
  - Decide what the answer should look like first, so you have something to compare it against
- **Read the diff as if a junior wrote it**
  - ”The tests pass" was never a review; "seems right" still isn't
- **Ask the model to argue against itself**
  - One prompt breaks the borrowed-confidence effect instantly
- **Notice when you're tired**
  - Surrender is a fatigue phenomenon; the fifth PR of the day gets a glance, not a review
- **Ask yourself: am I actually making the decisions?**

Controversial approach

(Source: https://x.com/mitchellh/status/2072738025344565262/)

Mitchell Hashimoto's 'I read the code' tweet broke the dev internet

(Source: https://daily.dev/posts/mitchell-hashimoto-s-i-read-the-code-tweet-broke-the-dev-internet-cqdrdqyfi/)

What we should do as a team

Habits we can adopt as a team

- **Make verification a hard exit criterion**
  - Every task ends in evidence (a test, a trace, a screenshot), not "it looks done"
- **Shrink the PRs**
  - The unit of review is the unit of comprehension; make the unit small enough to actually comprehend
- **Every PR description's "why" written by a human**
  - Can't be faked; if you can't explain it, you don't understand it
- **Add friction on purpose**
  - A design doc before generation, a checklist before deploy

Controversial approach

(Source: https://www.instagram.com/reels/Dbq4CDUOPqw/)

Closing thoughts

It is not an anti-AI propaganda! AI is a tool!

- Not an anti-AI talk
  - I use these tools daily and ship more because of them
- **The tools aren't the variable. The posture is.**
- The same model that hollows out one engineer's mental model sharpens another’s
  - The difference is thinking **with** it vs. instead of thinking
- Both postures ship code. Both look identical from the outside
- The difference shows up six months later, when something breaks and only one of them can fix it

Source materials

1. https://arxiv.org/abs/2601.20245 - How AI impacts skill formation
2. DOI: 10.31234/osf.io/yk25n_v1 - Thinking fast slow and artificial
3. https://pages.cs.wisc.edu/~remzi/Naur.pdf - Programming as Theory building
4. https://codemanship.wordpress.com/2025/09/30/comprehension-debt-the-ticking-time-bomb-of-llm-generated-code/
5. https://addyosmani.com/blog/comprehension-debt/
6. https://addyosmani.com/blog/cognitive-surrender/
7. https://www.anthropic.com/research/AI-assistance-coding-skills

Thank you for your attention!