Velocity Inflation: Your PR Volume Doubled and Nobody Shipped

Second Order Labs™ · June 2026

ABSTRACT

AI tooling makes pull requests pile up while deployment frequency flatlines. The bottleneck moved from writing code to understanding it, and most engineering metrics still measure the wrong end of the pipeline.

Keywords: DORA Metrics, Code Review, AI Coding, Technical Debt, Developer Productivity, Verification

> *"DORA metrics in the AI era reveal a paradox: PR volume is climbing, but deployment frequency is staying flat."*
>
> — Jeff Schinella, DORA Metrics in the AI Era: Why Deployment Isn't Faster

I. Your PR Volume Doubled and Nobody Shipped Faster

A team installs Claude Code. Pull requests climb. Commits multiply. The dashboard glows green with productivity. Then someone checks deployment frequency. It hasn't moved. Code piles up faster than anyone can decide it is safe to ship. This is not a tooling problem. It is a measurement problem. It breaks the apparatus engineering leaders use to know whether their teams are winning.

The standard diagnosis points to a review bottleneck. AI writes faster than humans read, so queues grow. True, but shallow. The deeper shift is that code generation flipped from a value-driver to a cost center. When implementation becomes effectively free, every line you produce is a liability you must comprehend and defend in an incident at 3 a.m. Metrics built for a world where writing code was the hard part now measure the wrong end of the pipeline.

> *The era of measuring how fast you produce code is over; the era of measuring how much code you can afford to not understand has begun. Call the gap between AI's drafting speed and a team's collapsing comprehension* ***Architectural Amnesia****: the rate at which an organization loses the coherent mental model of its own system. It is the metric nobody tracks and the one that will determine which teams survive recursive AI.*

II. The DORA Paradox: Why More Code Does Not Mean More Shipping

DORA metrics now lie by omission. PR volume and branch activity spike with AI tooling. Deployment frequency stays flat. Change failure rates creep upward. The acceleration is real but trapped in the wrong stage of the pipeline.

> *"DORA metrics in the AI era reveal a paradox: PR volume is climbing, but deployment frequency is staying flat." Jeff Schinella, DORA Metrics in the AI Era: Why Deployment Isn't Faster*

CircleCI's 2026 data sharpens the picture. Feature branch throughput is up while main branch throughput has fallen.[1] AI accelerates isolated execution but does nothing for the rate at which integrated value reaches users. AI-augmented teams generate dramatically more code artifacts, but the rate at which integrated value reaches users remains stubbornly flat.[2] The bottleneck has not vanished. It moved from writing code to deciding whether code is safe to merge.[3]

That relocation guts the predictive power of every throughput metric. Story points and lines of code assumed drafting was expensive, so counting drafts approximated counting effort. Once AI drops drafting cost to near-zero, those metrics measure the cheapest part of the work.[4] A team optimizing for velocity in this regime is optimizing for the production of liabilities. The faster the number climbs, the worse the underlying position is, because each merged diff adds to a codebase no human fully holds in their head.

Figure 1: AI accelerates the front of the pipeline while comprehension stays a human-speed activity at the end.

III. Verification Drag and the New Senior Engineer Burnout

The constraint inverted. The hard part of software is no longer generation. It is verification: reading and judging code rather than writing it.[5] AI raises the arrival rate of pull requests faster than teams can review them. This produces growing queues and larger batches, concentrating cognitive load on the few people who deeply understand the system.[6]

This breeds a burnout traditional metrics cannot see. It is not typing fatigue. Siddhant Khare described it precisely: exhaustion comes "not from writing code, from judging code. Hundreds of small judgments, all day, every day."[7] A BCG and UC Riverside study found that workers supervising AI agents reported significantly higher decision fatigue alongside a sharp rise in both minor and major errors.[8] The act of overseeing AI does not just tire reviewers. It degrades the quality of their oversight, which is the only safeguard left.

The dangerous coping mechanism is rubber-stamping. Faced with a massive AI-generated diff no human can fully hold in context, an overwhelmed reviewer approves it anyway.[6] The problem, as CodeRabbit framed it, is that "the volume of code has outpaced the amount of attention and time any human can give

it."[9] Throwing a second AI at the reading does not fix this. Comprehension was the point, not the keystrokes.[10]

Figure 2: Verification drag: the human checkpoint collapses under volume, and approvals become a formality.

IV. Architectural Amnesia: The Risk Nobody Measures

Here is where the review-bottleneck framing stops short. The real damage is not slow merges. It is the slow erosion of human understanding of the system itself. When AI takes over generation, review, and testing, no human maintains a coherent mental model of how the pieces fit. The organization develops **Architectural Amnesia**.

The symptom shows up first as a "Black Box Effect." Teams lose the deep model of their own architecture because they never wrote the implementation.[11] The consequence surfaces at the worst possible moment. When no human holds a coherent model of how the pieces fit, incident response degrades from engineering into guesswork.[12] You cannot debug a system you do not understand, and you cannot understand a system you never built.

This compounds because the incentives push the wrong way. It is now easier to prompt AI for new code than to read and refactor existing code, so teams add rather than fix. The result is software inflation: more code worth less per line, with maintenance debt injected faster than any traditional review cycle can absorb.[13] [14] Code ships faster, but bugs arrive later and debt compounds.[15]

V. Code-as-Liability Economics: Penalize Volume, Not Velocity

If generation is free, code is a liability, not an asset. That single reframe reorders every engineering metric. The premium moves entirely to architecture and specification. Those remain scarce and expensive while implementation no longer is.[10] The human checkpoint must shift upstream, from reviewing syntax to reviewing intent and design constraints.

A new metric category is forming to capture the friction the old ones hide. "Agent Efficiency" measures the gap between AI drafting speed and human review effort, replacing legacy throughput.[16] Analytics firms like Larridin extend DORA with Code Turnover Rate and Complexity-Adjusted Throughput, since Deployment Frequency and Lead Time have become misleading without context about AI code share and complexity.[17] The common thread: stop counting output, start pricing comprehension.

Old regime (generation is expensive)	New regime (verification is expensive)
Lines of code = productivity	Lines of code = liability surface
Velocity / story points	Agent Efficiency, Complexity-Adjusted Throughput
PR volume signals progress	PR volume signals review debt
Deployment Frequency (raw)	Code Turnover Rate, blended throughput
Reward more code	Penalize unowned code

The practical move most teams miss is to make code volume a cost line in the metrics, not a productivity indicator. Treat each merged diff as comprehension debt incurred. Pair this with Comprehension Audits: a deliberate check that some human can explain the architecture a merge touches, before it lands, not after the incident. If no one can explain it, the merge is debt regardless of how clean the tests look.

VI. What Happens When AI Writes the Spec Too?

Recursive AI is the endgame. Agents will write and review their own code, creating systems opaque to the humans nominally responsible for them.[12] At that point the human role shifts from creator to constraint-manager. The only metric that matters is whether a human can still specify and verify intent.

Finding people who can do that presents a cruel structural problem. AI automates the entry-level tasks that served as the apprenticeship for building judgment. Organizations hire seniors while automating juniors, causing the pipeline that produces the next generation of seniors to quietly collapse.[19] The same judgment Architectural Amnesia destroys at the system level is starved at the human level.[20]

Survival in a recursive AI environment requires measuring comprehension before you are forced to. Track Architectural Amnesia like you track uptime: which subsystems have a living human owner who can explain them, and which have gone dark. Run Comprehension Audits as a merge gate. Price code as a liability in your dashboards. Protect the apprenticeship path deliberately. Seniors who can supervise AI are the one resource the market consumes and refuses to replace. The constraint was never how fast you can write. It is how much you can afford to forget.

KEY FINDINGS

- AI tooling speeds up feature-branch work but slows main-branch throughput, so PR volume climbs while deployment frequency stays flat.
- A BCG and UC Riverside study found that workers supervising AI agents reported 33% higher decision fatigue and 39% more major errors.
- Story points and lines of code measure drafting effort, and that's the one thing AI just made nearly free.
- The Black Box Effect leaves teams unable to debug systems they never wrote, so incident response turns into guesswork.
- Organizations hire seniors and automate juniors, which drains the apprenticeship path that produces the next generation of seniors.

REFERENCES

[1] AI Is Breaking Code Review: How Engineering Teams Survive the PR Bottleneck (CircleCI 2026 data). https://circleci.com/resources/2026-state-of-software-delivery/

[2] SAFe Sprint Cadence for AI Teams: The Dual-Rhythm Architecture, Agility at Scale. https://agility-at-scale.com/safe/ai-enabled-safe/safe-sprint-cadence-for-ai-teams-the-dual-rhythm-architecture/

[3] AI Is Breaking Code Review: How Engineering Teams Survive the PR Bottleneck. https://blog.codacy.com/ai-breaking-code-review-how-engineering-teams-survive-pr-bottleneck

[4] Why Are Agile Metrics Broken for AI Coding Teams?, YouTube.

[5] AI Code Review Bottleneck: Fix PR Congestion in 2026, Metacto. https://www.metacto.com/blogs/code-review-bottleneck-ai-development

[6] AI Coding Made Code Review the Bottleneck. Now What?, Yuval Yeret.

[7] AI fatigue is real and nobody talks about it, Siddhant Khare. https://siddhantkhare.com/writing/ai-fatigue-is-real

[8] AI psychosis is real, and you probably have it, Vellum (citing BCG and UC Riverside). https://www.vellum.ai/blog/ai-psychosis-is-real

[9] The real bottleneck in code review isn't reviewing code, it is understanding it, CodeRabbit. https://www.coderabbit.ai/blog/bottleneck-in-code-review-is-understanding-intent

[10] How long before we stop reading the code?, The New Stack. https://thenewstack.io/future-of-code-reviews/

[11] Building Trust in AI-Generated Code Through Continuous Testing, Testkube. https://testkube.io/blog/building-trust-in-ai-generated-code-through-continuous-testing

[12] The Ouroboros Machine: When AI Reviews Its Own Code, SmarterArticles. https://smarterarticles.co.uk/the-ouroboros-machine-when-ai-reviews-its-own-code

[13] The Neuro-Coder (Part 5): Beyond DORA (Why Velocity is Meaningless), Sapient Coffee.

[14] What Is Technical Debt in AI Coding? Types & Impact Explained, Janea Systems. https://www.janeasystems.com/blog/technical-debt-ai-coding-types-impact

[15] Developers won't work without AI anymore. The research says it might be making them worse, TNW. https://thenextweb.com/news/developers-refuse-work-without-ai-coding-productivity-paradox

[16] Agent Efficiency: AI Era Agile Metrics, Scrum Day India. https://agileleadershipdayindia.org/blogs/agentic-ai-agile-project-office/ai-agile-agent-kpis-metrics.html

[17] Why DORA Metrics Break in the AI Era, Larridin. https://larridin.com/developer-productivity-hub/why-dora-metrics-break-ai-era

[18] Anthropic Co-Founder on our wild 'recursive' AI future, TechRadar. https://www.techradar.com/ai-platforms-assistants/you-would-be-able-to-say-to-it-make-a-better-version-of-yourself-and-it-just-goes-off-and-does-that-completely-autonomously-anthropic-co-founder-on-our-wild-recursive-ai-future

[19] Junior Developer Pipeline AI Crisis: The Narrowing Pyramid, Cloud Perspectives. https://cacm.acm.org/opinion/redefining-the-software-engineering-profession-for-ai/

[20] Microsoft's Russinovich and Hanselman Warn AI Is Hollowing out the Junior Developer Pipeline, InfoQ. https://www.infoq.com/news/2026/04/junior-developer-pipeline-crisis/
