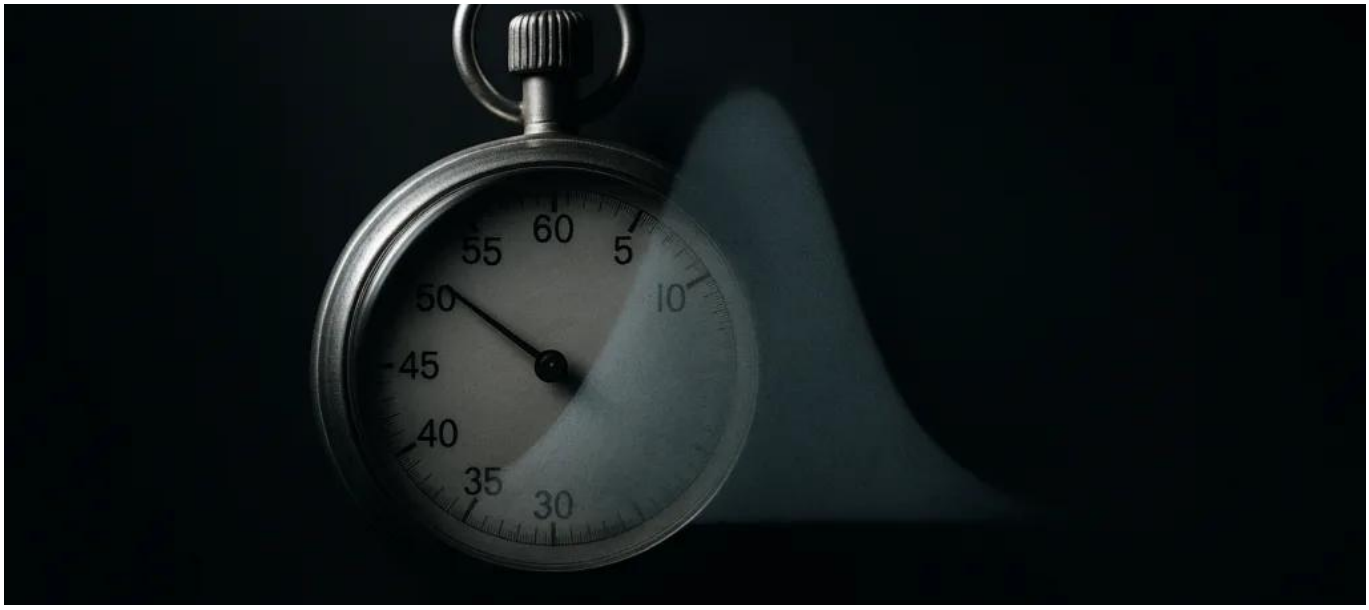

The METR 19% Slowdown Was Never About AI Being Slow: It Was About the 39-Point Gap Between Feeling and Stopwatch

The most-cited study in AI coding just reversed its own sign, and the number that mattered was never the 19%. It was the 39-point gap between what developers felt and what the clock recorded.

METR's July 2025 randomized trial measured 16 experienced open-source contributors, people with roughly five years in their own repositories, working 246 real issues on codebases that often exceeded a million lines. With AI access they were 19% slower. The confidence interval ran from 2% to 39% slower, no scenario showed a speedup, and the result went straight into every skeptic's slide deck. Then on February 24, 2026, METR published an update: the 10 returning developers now measure 18% faster (interval from 38% faster to 9% slower), while 47 newly recruited developers measure 4% slower (interval from 15% slower to 9% faster). Both new intervals cross zero. METR said so itself, and added

the caveat that because of selection effects in the experiment, its data is “only very weak evidence for the size of this increase.”

Almost nobody noticed the flip, because both camps had already finished quoting the part they liked.

What 16 developers and 246 issues could and could not tell you

The original METR result deserved its attention. It was a real RCT on real work rather than a benchmark harness or a self-report survey, and its design was unusually honest about the thing most productivity research avoids: the codebase people actually maintain. High-familiarity contributors on million-line repositories are the hardest possible case for AI assistance, and that is why the finding was interesting. If you want to know whether an assistant helps a senior engineer on a mature system, that is the population you study.

What the study could not support was the sentence everyone extracted from it. “AI makes developers 19% slower” was a point estimate from 16 people with a 37-point-wide interval, and it got read as a law of nature. The 2026 update is the same design running longer with more participants, and the direction moved. Small-n research does that when you add data. What bothers me is that an industry spent eight months arguing about a mean while the interval was screaming that the mean was not the finding.

WHAT GETS QUOTED	WHAT METR NOW REPORTS
METR proved AI coding tools make experienced developers 19% slower.	The February 24, 2026 update puts 10 returning developers at 18% faster (38% faster to 9% slower) and 47 new developers at 4% slower (15% slower to 9% faster). Both intervals cross zero, and METR calls its own data “only very weak evidence for the size of this increase.”

The perception gap survived the sign flip intact

Before the trial, developers forecast that AI would make them 24% faster. After doing the work, having lived through it, they estimated 20% faster. The stopwatch said 19% slower. That is a 39-point gap between felt and measured, and the experience of doing the work did not correct it: the prediction and the retrospective estimate were four points apart.

It is not a quirk of 16 people, either. It replicates at scale. DORA 2025 found AI adoption among software professionals rose from 76% in 2024 to 90% in 2025, with a median of two hours per day spent working with AI tools. In the same report, over 80% of respondents say AI enhanced their productivity and 59% report increased code quality. In that same report, the measured relationship between AI adoption and delivery instability stayed negative. The self-reports and the telemetry are describing the same teams and disagreeing about what happened to them.

Stanford's Yegor Denisov-Blanch has the largest measured sample I am aware of: roughly 100,000 developers across 600-plus companies. His finding is that apparent gains of 30 to 40% compress to somewhere around 15 to 20% net once rework is subtracted. Rework, defined as code changed again within three weeks of being introduced, increased 2.5x in some AI-adopting cohorts, with code quality down 9%. The delivered output is real. Roughly half of it gets eaten by the second pass nobody counted.

One average number cannot exist for a distribution with no center

My take: the productivity debate is broken because it is trying to compute a mean over a distribution that has no center. Denisov-Blanch's four quadrants make this concrete. Low-complexity greenfield work shows 30 to 40% gains. High-complexity brownfield work shows 0 to 10%, with some teams net negative. Those are different regimes, not noisy estimates of one underlying truth.

Now look at METR through that lens. Sixteen senior maintainers, five years of context each, million-line repositories: that is the high-complexity brownfield corner of Denisov-Blanch's grid, where he measures 0 to 10% and sometimes negative. METR's original 19% slowdown is

a plausible draw from that cell. The 2026 returning-developer estimate of 18% faster is a plausible draw from a cell where people have had eight more months of practice with the tools. Neither number tells you what will happen on your team, because your team sits somewhere else on the grid and you probably have not located yourself.

DORA's own framing is the closest thing to a usable model. It calls AI an amplifier: the tools magnify an organization's existing strengths and weaknesses. That explains the 2024-to-2025 reversal in its own data better than any story about model quality. DORA 2024 measured that every 25% increase in AI adoption cut delivery throughput 1.5% and delivery stability 7.2%. DORA 2025 reversed the throughput half, reporting "a positive relationship between AI adoption on both software delivery throughput and product performance," and left instability negative. Throughput is what a good pipeline can absorb. Instability is what a weak pipeline converts into incidents. Same tools, different substrate.

Where I land: the 19% number and the 18%-faster number are both correct and both useless as decision inputs. The stable, replicated finding across METR, DORA, and Stanford is a systematic gap between perceived and measured effect, always in the same direction, and it does not close with experience. If your AI investment case rests on developer sentiment, you are budgeting against the one variable every study agrees is biased.

I wrote something similar about the GitLab survey where 78% of developers reported coding faster while delivery metrics did not move. The pattern keeps showing up in the same shape: strong subjective acceleration, flat or worse system-level outcomes, and an argument about whether the tools work that never gets around to asking which part of the pipeline absorbed the difference.

The strongest argument against me is that the stopwatch measures the wrong interval

Perception may not be wrong here, and I should not be so confident the clock wins.

If AI removes the cognitive cost of a task without removing the clock time, a developer can finish the same issue in the same duration and have capacity left for the next one. Time-per-issue would show nothing. Sustained daily output might show something. DORA's two-hours-per-day median suggests a large fraction of the working day now runs through these

tools, and no per-task RCT captures what that does to the eighth hour.

Then there is the fact that METR's returning developers got faster. That could be the selection effect METR warns about, and I think some of it is. It could also be genuine skill accumulation: eight months of learning when to delegate and when to close the assistant. If that is what happened, then every 2025 measurement is a snapshot of an untrained population, and the honest reading is that we do not yet know the steady state. METR's own caveat cuts in this direction as much as mine.

Denisov-Blanch's rework metric also has an assumption inside it. Code changed within three weeks is not automatically waste. On fast-iterating product surfaces, rapid revision is the process working. A 2.5x rework increase alongside a 9% quality decline is suggestive, and I read it as mostly cost, but the metric cannot separate churn from iteration on its own.

What survives all three objections is the gap itself. You can argue about the sign of the productivity effect. You cannot get from "80% report gains, 59% report better quality" to "measured instability got worse" without conceding that self-report and measurement are tracking different things.

What would move me off the multi-modality argument

I expect METR's sample to keep widening its intervals rather than converging on a number, because the underlying distribution is genuinely multi-modal across task complexity. If a future update reports a tight interval that excludes zero in either direction on a sample above 50 developers, I am wrong about the multi-modality and there is a real average effect to argue about.

I expect DORA 2026 to still show instability negative even if throughput stays positive, for the same reason it did in 2025: instability is downstream of review capacity, test coverage, and deployment discipline, and those have not changed as fast as generation volume. If instability flips positive alongside throughput, the amplifier model is incomplete and something about the tools themselves improved the failure side, which I would want to understand.

I expect the perception gap to persist. This is the falsifiable one I care most about. If a

study with a measured baseline finds developer self-estimates landing within, say, 10 points of stopwatch time, then the gap was a 2025 artifact of novelty and the sentiment data becomes usable. Until then, treat "the team says it is faster" as a morale signal rather than a throughput signal.

The practical consequence is unglamorous. Stop asking whether AI makes developers faster. Measure your own rework rate with a fixed window, watch your change failure rate, and instrument the quadrant you actually work in: mature codebase or new one, high complexity or low. The industry-average number does not exist, and the four studies above have now spent two years demonstrating that from four directions. Related: the same measurement problem shows up in evaluation, where the same scaffold scores 65% or 74% on SWE-bench depending on harness details.

WHAT TO DO WITH THIS

The 19% slowdown and the 18% speedup are the same study telling you it cannot produce a single number. The finding to act on is the 39-point gap between what your developers feel and what your systems record: instrument rework and change failure rate in your own codebase before you budget against sentiment. If you want help deciding what to measure and what to ignore, that is a short conversation. Book a call →

WRITTEN BY

Artur Markus

Software engineer in Basel building AI infrastructure and agentic systems, with a background in pharmaceutical automation, GxP compliance, and validation. I help teams turn AI from a chatbot into a reliable operational layer.

Book a meeting →