ANNUAL RESEARCH REPORT

The State of AI Software Productivity 2026

What the data actually shows for software developers and business leaders - the studies, the business results, and how to measure it.

- 2026 Edition
- 2026-08-05
- 18 min read

AI-assisted software development is no longer a question of whether teams will adopt it - it is a question of whether the results are real, measurable, and worth the money. This report answers that question with the 2026 evidence, in one place: the developer-level studies, the business-level results, the measurement method that connects them, and the honest caveats most AI content skips.

Executive summary

Is AI actually making software developers more productive? Yes - but the honest answer has three parts.

+40.5%

More pull requests completed in high-usage weeks, holding coding time constant. The most rigorous causal study to date.

Microsoft, 16,223 engineers, dose-response

20-40%

The typical real-work range of measured gains - and occasionally double for greenfield work, occasionally negative for senior teams on legacy code.

Across credible 2026 studies

-$23M/yr

Reported savings from AI-assisted coding at a single enterprise. Rakuten cut feature time-to-market 79%. Amazon reports $260M/yr from one modernization program.

One insurer, Opsera customer story

-19%

Experienced open-source developers on mature codebases were slower with AI. Gains depend on the team and the task.

METR RCT, 16 senior devs

78%

Of leaders report more production incidents even as 94% rate AI code higher quality at review. Speed is up, stability is down.

New Relic 2026

86% vs 60%

Satisfied with AI tools - yet ~60% report under one hour a week of actual time saved. Happy developers are not the same as productive developers.

BNY Mellon, 2,989 devs

+55%

More likely to see major efficiency gains with full AI governance. Governance is the multiplier.

Black Duck, 831 engineers

400%

Reported 3-year ROI, under 6-month payback. DORA's own model: 39% first-year ROI for a 500-person org.

Forrester TEI, GitLab Duo (commissioned)

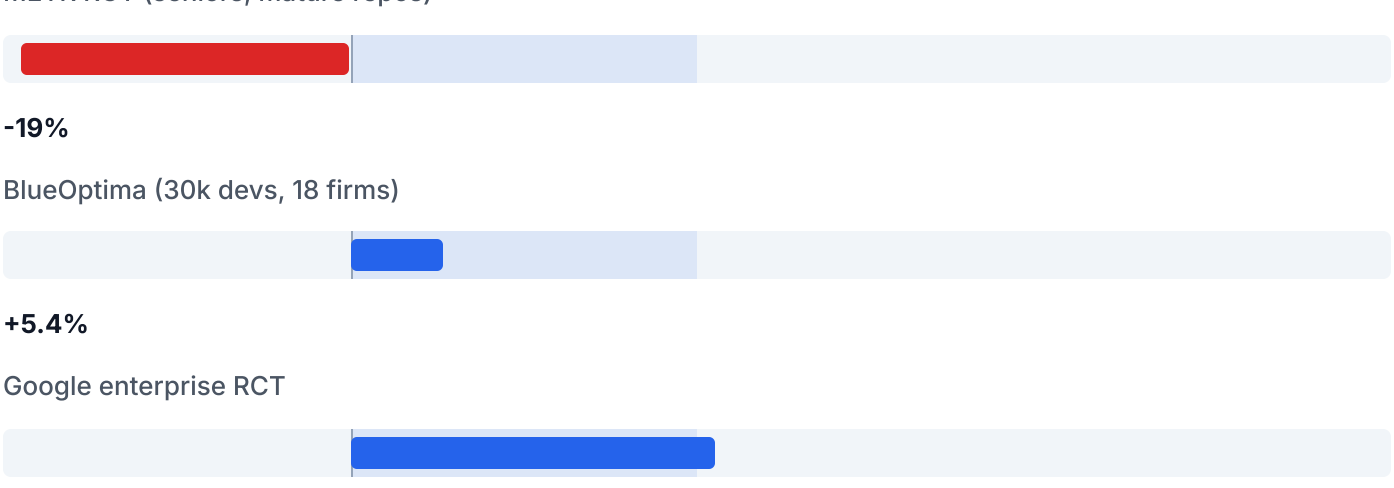
> **The one-line version for a decision maker**
>
> AI-assisted development delivers real, measured gains of roughly 20-40% on most work - and the gains only hold if you measure them properly and govern the process. AI is an amplifier, not a replacement.

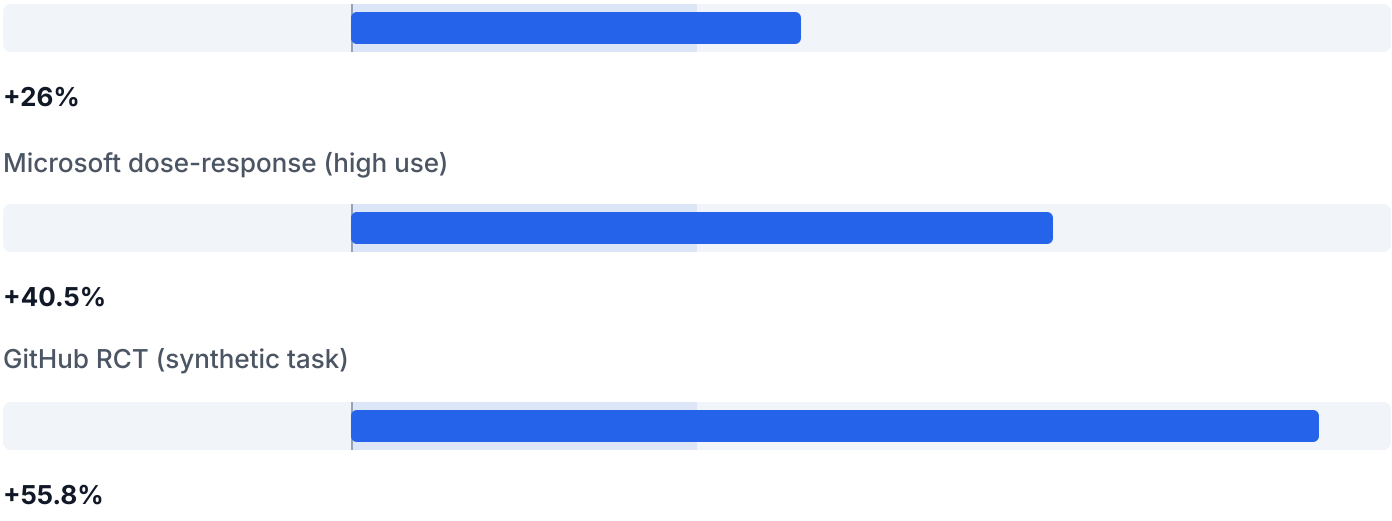
Part 1: The developer-level evidence

For engineering leaders. The 2026 evidence base is dramatically better than 2023. Early results came from small lab experiments; 2026 has large, causal, peer-reviewable studies.

STUDY	SCALE / METHOD	RESULT	SOURCE
Microsoft dose-response (2026)	16,223 engineers, 43 wks, within-engineer	**+40.5% PRs** at high usage; +21% low; +39% moderate	independent
Microsoft CLI rollout (2026)	Tens of thousands, rollout analysis	**+24% PRs merged**, sustained; +50.1% at 5+ days/wk	independent
Microsoft field RCT (2025)	4,867 devs, 3 companies	**+26% tasks completed**	independent
Google enterprise RCT (2024)	96 engineers, real task	**~21% faster**	independent
UK public sector trial (2025)	2,500 licences	**56 min/day saved** (~28 days/yr)	independent
GitHub Copilot RCT (2023)	95 devs, synthetic task	**55.8% faster**	vendor
BlueOptima (2026)	30,000+ devs, 18 enterprises	**+5.4%**, up to +20% heavy users	analysis
Black Duck survey (2026)	831 engineers	92% improved; **8 hrs/wk saved**	vendor survey
METR RCT (2025)	16 senior devs, mature repos	**-19% (slower)**	independent

Measured productivity change across studies





Shaded band = typical real-work range (20-40%). Synthetic greenfield tasks show the largest gains; fixed-outcome real tasks on mature codebases the smallest - or none.

What actually flips the result

- **Methodology.** Synthetic greenfield tasks show the biggest gains (GitHub's 55.8%). Real tasks on mature codebases show less, or none (METR's -19%). The within-engineer dose-response design is the most credible because it compares each engineer against their own baseline.
- **Experience level flipped in 2026.** 2023-2025 studies said juniors gained most from autocomplete. Opsera's 2026 benchmark of 250,000+ developers found **seniors capture nearly 5x the gains** - the agentic tools reward skill at directing and reviewing AI output. This changes who you hand the tools to first.
- **Task type.** Tedious, well-specified work - migrations, refactors, boilerplate - is where the gains are largest (Amazon's Java modernization: ~50 dev-days per app down to hours). Open-ended architecture and design work gains the least.

The measurement trap everyone keeps hitting

BNY Mellon surveyed 2,989 developers: **86% are satisfied** with Copilot, but **~60% report under one hour a week of time savings**, and satisfaction barely correlates with savings (r = 0.34). Meanwhile GitHub's original study found developers *underestimated* their gain (perceived 35% vs measured 55.8%), and METR found developers and expert observers

alike *overestimated* theirs. Perception diverges from reality in both directions. That is why the rest of this report is about measurement, not anecdotes.

Part 2: The business-level evidence

For decision makers. The developer studies tell you it is real. These named-company results tell you what it is worth. Insurance leads because it produces the cleanest hard-dollar stories in the most regulated, risk-averse sector - the one a Chief Underwriter will recognize.

Insurance

COMPANY	RESULT	SOURCE
Leading insurer (anon)	**$1.9M/month (~$23M/yr) savings**; 55,000+ dev-hours saved monthly; 30% faster delivery; 45% faster onboarding	Opsera story
Unum (US)	**$20M expected benefits** from AI-rewriting 1.5M lines of COBOL; claims screens cut 7 to 1; ~7,000 man-hours/yr	Computer Weekly
Markerstudy (UK)	Assessed **6M lines of RPG code in 4 weeks**; path to **10M/yr savings**	Ensono story
NOSI (social insurance)	**94% faster** COBOL analysis (8 hrs to 30 min); 79% faster understanding of complex apps	IBM story

Banking & fintech

COMPANY	RESULT	SOURCE
National Australia Bank	**40% of production code AI-generated**; 18% productivity boost; 3,000+ devs	AWS story
PayPal	Java upgrade of **3,000 apps in 2 months** (was 8-12 months); 4-sprint projects to 1; ~40% more capability delivered in 2026 vs 2025	Cursor story
WEX (fintech)	~30% higher productivity; **~60% ROI on licenses**; **99% reduction in deployment cycle time**	vendor story

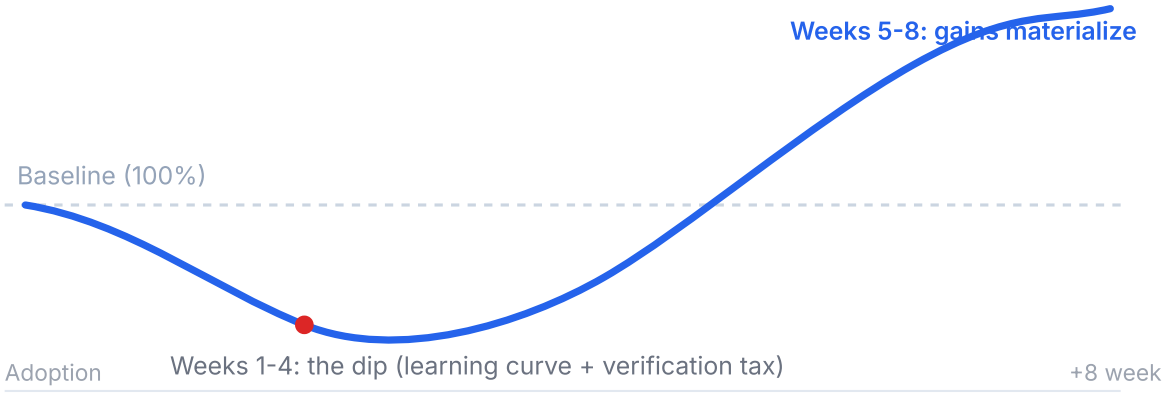
Telco, tech, retail

COMPANY	RESULT	SOURCE
TELUS	**500,000+ hours saved** across 57,000 team members; shipping 30% faster; 13,000+ internal AI solutions	Anthropic report
Rakuten	**Time-to-market cut 79%** (24 days to 5 per feature); **97% reduction in critical errors**; releases every 2 weeks instead of quarterly	Anthropic story
Doctolib	Legacy testing infra replaced in **hours instead of weeks**; onboarding weeks to days	Anthropic story
General Motors	Build times **4-6 hrs to 27 min**; COBOL/Fortran modernization that would have taken a year externally	GitHub story
Trimble	**1,000 hours/day saved** across 2,000 devs; 30 min/day each	GitHub story
Amazon	**$260M/yr and 4,500 developer-years** from Java modernization of 30,000 apps	AWS story
Motorway	**4x engineering output**; 1M LOC/month; 250% more deployments	Computer Weekly

The credibility layer: whitepapers and TEI studies

STUDY	RESULT	SOURCE
GitLab Duo + Forrester TEI (Jul 2026)	**400% ROI, $7.5M NPV over 3 years, payback under 6 months**; 80% faster onboarding; 75% faster migration	commissioned
DORA ROI model (Jan 2026)	**39% first-year ROI**, ~8-month payback for a 500-person org	independent
Anthropic State of AI Agents (2026)	**80% of orgs report measurable economic returns** from AI agents	vendor

The J-curve: budget the dip, judge the quarter



DORA 2026 research: most organizations see a temporary productivity dip in the first weeks as workflows adapt, before gains materialize. Judge the result over a quarter, not a month.

The five outcomes that repeat

Every credible business case study lands on the same five outcomes. If a vendor story does not move one of these, treat it as marketing:

1. **Time to market** - Rakuten -79%, PayPal 4 sprints to 1, Motorway +250% deploys
2. **Hard dollar savings** - $23M/yr (insurer), $20M (Unum), 10M/yr (Markerstudy), $260M/yr (Amazon)
3. **Legacy modernization** - the single biggest recurring story (COBOL, RPG, Java)
4. **Onboarding & learning curve** - Doctolib weeks to days, GitLab -80% onboarding
5. **Capacity / output** - PayPal +40% capabilities, Trimble 5x components, Motorway 4x

Part 3: How to actually measure it

The bridge - for both readers. This is the part nobody else publishes: how a developer metric becomes a business result, in a way a CFO or underwriter can defend. It is the method we use with clients.

Step 1 - Baseline before you adopt

Instrument the DORA four from your ticketing system and git, before anyone touches an AI tool: deployment frequency, lead time for changes, change failure rate, and time to restore service. Add review-queue depth and the fully-loaded cost per developer. Most small teams can pull this in an afternoon. This baseline is your "before" number; without it you have no claim.

> **Coding is only part of an engineer's day**
>
> GitHub CEO Thomas Dohmke notes that in most companies, developers write code for only about two or three hours a day. The rest is spent reviewing coworkers' code, fixing warnings and linter errors, clearing security backlogs, and other non-coding work. If your baseline only tracks commits and pull requests, you are measuring a minority of the work - and missing most of the friction.

Step 2 - Run a small controlled test

4-6 weeks, one pod with the tool vs one comparable without - or before/after on the same kind of task. The UK government (2,500 licences, 56 min/day) and Google (96 engineers, +21%) used this design. For a larger team, use the **within-engineer dose-response** design: compare each engineer's high-usage weeks against their own zero-usage weeks. This controls for confounding without needing a control group, and it is what makes a number survive cross-examination.

Step 3 - Measure output AND quality

Speed is easy to fake; stability is not. Track change failure rate, rework, bugs found in review, and rollbacks alongside throughput. GitClear's analysis of 211M changed lines is the cautionary tale: code cloning up ~4x, refactoring down, and short-term churn up since AI adoption. Fast code that needs constant rework is not a productivity gain.

Step 4 - Convert hours to dollars

Loaded cost per developer x hours saved is the number a decision maker can act on: the UK trial found 56 min/day (~28 working days/year), Black Duck ~8 hrs/week, Google/CME ~10.5 hrs/month. A 10-person team saving 5 hours/week each at a $120K loaded cost is roughly **$300K/year of reclaimed capacity**. That is the number to argue with.

Step 5 - Decide, don't drift

After the test window, use the data to expand, right-size licenses (one SaaS found $4,560/year of waste in 20 unused seats; Opsera found 21% of licenses underutilized), or fix the system before scaling. Set governance guardrails at the start - governed teams are 55% more likely to see major gains.

> **Triangulate - one number is never enough**
>
> Combine delivery data + a short survey + the one experiment. The survey catches satisfaction and cognitive load the data misses; the experiment anchors causation; the delivery data grounds it in the real system. Never judge on self-report alone, and never judge in the first 4 weeks - the J-curve means you will see a dip before the gains.

Part 4: The honest caveats

Read this before you write the cheque.

- **Most business numbers are vendor-published.** GitHub, AWS, Anthropic, and Opsera publish their customers' wins. They are directionally real but selection-biased - nobody publishes a failed pilot. The independent anchors are the Microsoft studies, BlueOptima, BNY Mellon, DORA, and press-reported cases (Unum, Motorway). We label sources throughout for this reason.
- **Throughput is up, stability is down.** New Relic: 94% rate AI code higher quality at review, yet 78% report more production incidents and 86% more senior firefighting.

DORA's model has change failure rising from 5% to 6%. AI accelerates delivery; it also amplifies whatever is broken in your process.

- **The security cost is real.** Opsera found AI-generated code introduces 15-18% more security vulnerabilities, and 64-73% of teams are concerned. Review becomes the new bottleneck - AI PRs wait 4.6x longer in review.
- **AI is an amplifier, not a fix.** DORA's finding across two years: AI magnifies the strengths of high-performing organizations and the dysfunctions of struggling ones. If your team already ships badly, AI will ship badly faster.

> **What is still hype**
>
> Claims of 10x or 20x productivity for typical teams (that was a no-code tool, and even Anthropic says developers can only fully delegate 0-20% of tasks), "AI will replace developers" (no credible 2026 study supports it), and any single-number productivity answer.

Part 5: The decision-maker playbook

A 4-6 week evaluation any business can run.

1. **Pick one bounded workflow** - a recurring feature type or migration, not "everything."
2. **Baseline it now** - the DORA four + review queue + cost per developer, before tools.
3. **Pilot one tool with one pod** for 4-6 weeks; keep a comparable pod as control.
4. **Track speed AND quality** - throughput, change failure rate, rework, bugs.
5. **Convert to dollars** - hours saved x loaded cost; compare against tool spend.
6. **Budget the J-curve** - ignore the first 3-4 weeks of data.
7. **Decide with the data** - expand, right-size licenses, or fix the system first.
8. **Govern from day one** - define what code can ship without review, who can deploy, and how AI-generated code is reviewed.

> **The honest expectation to set with your leadership**
>
> Roughly 20-40% more output on most work within a quarter, with stability held or improved - not a 10x overnight story. If anyone promises you 10x, ask for their measurement.

Methodology & sources

How this report was assembled: we compiled the 2025-2026 evidence base - developer-level studies and business-level results - then ranked it by methodological strength and labeled every source by independence. We did not run our own survey this year; future editions will add original data. Numbers from vendor customer stories are reported as claimed by the vendor and cross-checked against trade press where possible.

Source tags: independent (peer-reviewable / academic / government / Google DORA) - vendor (published by the tool vendor) - commissioned (paid third-party study) - analysis (independent data-analysis shop).

Developer-level sources

1. Microsoft, GitHub Copilot dose-response analysis (2026). arXiv:2606.00438. INDEPENDENT. arxiv.org/html/2606.00438v1
2. Microsoft, Claude Code / Copilot CLI rollout (2026). arXiv:2607.01418. INDEPENDENT. doi.org/10.48550/arxiv.2607.01418
3. Microsoft Research, Effects of Generative AI on High-Skilled Work (2025). INDEPENDENT. microsoft.com/en-us/research
4. Google, enterprise RCT (2024). arXiv:2410.12944. INDEPENDENT. arxiv.org/html/2410.12944
5. Peng et al., GitHub Copilot RCT (2023). arXiv:2302.06590. VENDOR. arxiv.org/abs/2302.06590v1
6. METR, experienced OSS developer RCT (2025). arXiv:2507.09089. INDEPENDENT. doi.org/10.48550/arxiv.2507.09089

7. Black Duck / UserEvidence survey (Mar 2026). VENDOR SURVEY. blackduck.com
8. Opsera AI Coding Impact 2026 Benchmark. VENDOR. opsera.ai
9. GitLab AI Accountability Report (Jun 2026). VENDOR. gitlab.com
10. New Relic, 2026 State of AI Coding. VENDOR. newrelic.com
11. BNY Mellon, Beyond the Commit (2026). arXiv:2602.03593. INDEPENDENT. arxiv.org/html/2602.03593
12. BlueOptima, Impact of GitHub Copilot (2026). ANALYSIS. blueoptima.com
13. GitHub Copilot Impact Dashboard. VENDOR. github.com/resources/insights
14. Anthropic, Agentic Coding Trends 2026. VENDOR. anthropic.com
15. Anthropic, State of AI Agents 2026. VENDOR. anthropic.com
16. Google DORA, State of AI-Assisted Software Development (2025). INDEPENDENT. research.google
17. Google DORA, ROI of AI-Assisted Development (Jan 2026). INDEPENDENT. infoq.com
18. GitClear, AI Code Quality 2025. ANALYSIS. gitclear.com
19. UK Government Digital Service, AI coding assistant trial. INDEPENDENT. gov.uk
20. Aalto University, Copilot adoption case study (2025). ACADEMIC. aaltodoc.aalto.fi

Business-level sources

21. Opsera customer story - leading insurer. VENDOR. opsera.ai/customer-stories
22. Computer Weekly - Unum COBOL rewrite. PRESS. computerweekly.com
23. Ensono - Markerstudy. VENDOR. ensono.com
24. IBM - NOSI. VENDOR. ibm.com
25. AWS - National Australia Bank. VENDOR. aws.amazon.com
26. Cursor - PayPal. VENDOR. cursor.com/blog/paypal
27. Applied - WEX. VENDOR. theapplied.co
28. Anthropic - Rakuten. VENDOR. claude.com/customers/rakuten
29. Anthropic - Doctolib. VENDOR. claude.com/customers/doctolib

30. GitHub - General Motors. VENDOR. github.com/customer-stories
31. GitHub - Trimble. VENDOR. github.com/customer-stories/trimble
32. AWS - Amazon Q / Amazon. VENDOR. aws.amazon.com/blogs
33. Computer Weekly - Motorway. PRESS. computerweekly.com
34. GitLab + Forrester TEI (Jul 2026). COMMISSIONED. about.gitlab.com
35. Coriven - Copilot ROI unit economics. VENDOR. coriven.ai
36. Anthropic analysis - TELUS. VENDOR. udit.co

Measurement sources

37. DORA / SPACE / DevEx comparison. getdx.com/guide
38. SPACE framework (ACM CACM). cacm.acm.org
39. Datadog - measuring dev experience in the AI era. datadoghq.com
40. Xebia - rolling out AI coding assistants. xebia.com

Practitioner perspectives

41. Dohmke, T. (2026, Mar). GitHub CEO: Our devs barely write code. No Priors / YouTube. VENDOR. youtube.com/watch?v=3gEhfsenzTk

Recommended citation

RoboticForce. (2026, August). *The State of AI Software Productivity 2026*. https://roboticforce.io/reports/state-of-ai-software-productivity

Disclosure: this report was compiled by RoboticForce, an AI implementation firm. We sell implementation and measurement services; we do not sell AI coding tools. Where we cite a vendor, we flag it. Our interest is that you measure before you buy.