nirmata

MEASUREMENT MODEL AND REFERENCE ARCHITECTURE

AI ROI for **engineering**

A measurement model for the AI coding spend already on your books, and a reference architecture for tying every dollar of it to work that shipped.

WHAT MOST ORGANIZATIONS SEE	REQUIRED TO MEASURE ROI		
01 **Spend** invoice	02 **Request** gateway	03 **Change** pull request	04 **Outcome** merged and standing

SECTION 01

Executive summary

In October 2026 Anthropic made public its S-1 registration statement. It reports Net Revenue Retention of approximately 900% as of 30 September 2026: in the latest three months, the same customers paid Anthropic about nine times what they paid in the same three months a year earlier, excluding fixed-price Claude Pro and Max subscribers. Customers spending $100,000 or more a year grew from more than 950 to more than 11,900 in twelve months. [1] Across the market, coding is the largest single category of departmental AI spend — 55% of it in 2025. [2]

> **~900%**
>
> Net Revenue Retention reported in Anthropic's S-1, as of 30 September 2026. Same customers, same quarter, one year apart.
>
> ANTHROPIC, PBC · FORM S-1 · P. 134

The filing attributes that growth to "significant usage expansion as customers realize the value of deploying Claude across their organizations." That is the vendor's account of the value. The buyer's account is much harder to find.

Nine times the spend is a fact. Nine times the output is not — and no engineering organization we work with can show it. Most cannot yet show any multiple at all, because they are not measuring the thing the money buys.

That is not a model problem. The models are good and the tools built on them are good. The problem is that the unit everyone manages is the wrong one. Finance manages seats and invoices. Platform teams manage tokens and rate limits. Vendors report acceptance rates and "AI-assisted" lines of code. None of those is a unit of value. A token tells you what was consumed. It says nothing about what was produced.

> "The token is the billing unit, not the value unit."
>
> FinOps Foundation, Tokenomics, June 2026

In the twelve months to June 2026, every major coding-agent vendor moved from flat seats to usage-based billing. The variable bill landed on organizations with no way to attribute it. One large engineering organization was reported to have spent its entire 2026 AI budget by April, driven by coding-agent usage. [3] Gartner now forecasts that AI coding costs will exceed the average developer's salary by 2028. [4]

This paper proposes one management unit: **AI cost per accepted outcome** — attributed AI spend divided by pull requests that merged and were still standing two weeks later, read against guards for quality and review load. It is unit economics, applied to a workforce in which part of the work is now done by agents billed by the token.

The unit is easy to define and hard to compute, for three reasons. Spend must be attributed from the token to the developer, the session, the repository and the change. Output must be read against a baseline honest enough to survive a skeptical CFO. And spend follows a power law, not a bell curve, so the average hides almost everything that matters.

This paper is for the engineering leaders who own both the AI budget and the delivery commitments it is supposed to accelerate. It assumes you have rolled out at least one coding agent. It does not assume you know what that rollout has returned. In the organizations we work with that number rarely exists, and producing it is usually the first useful thing any of this yields.

HOW TO READ THE NUMBERS IN THIS PAPER

PUBLISHED

A primary source — a filing, a vendor's own announcement or documentation, a peer-reviewed or methodologically published study.

REPORTED

A secondary account of a primary source we could not read directly, such as press coverage of paywalled reporting or of a conference talk. Marked where it appears.

OBSERVED

Drawn from AIControls running in Nirmata's own engineering organization, rather than from anything published. One organization, so an example and not a benchmark. The method is in Appendix A.

ILLUSTRATIVE

A constructed example that shows the arithmetic. Never customer data.

These are not interchangeable, and an ROI argument that blurs them is asking you to accept a conclusion you cannot check.

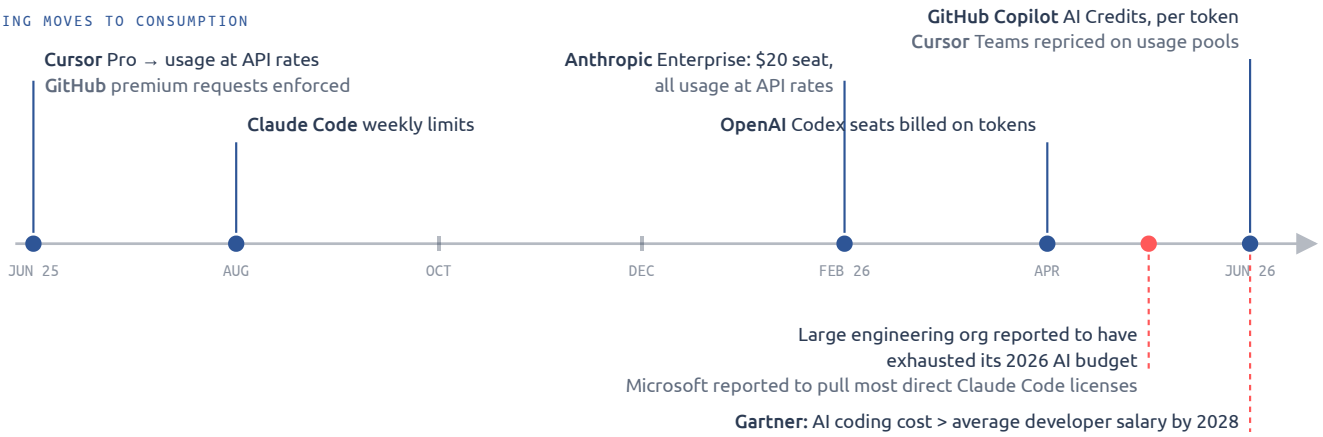
SECTION 02

The bill changed shape

For most of 2024 an AI coding assistant was a seat — $19 or $39 per developer per month, budgeted alongside the IDE license and forgotten. Between June 2025 and June 2026 that stopped being true, vendor by vendor.

TWELVE MONTHS, FOUR VENDORS, ONE DIRECTION

PRICING MOVES TO CONSUMPTION



WHAT FOLLOWED

Every vendor moved from pricing access to pricing consumption, and every one gave the same reason: an hour-long autonomous coding session costs orders of magnitude more than a chat question, and a flat price could not hold. Sources 3–10.

VENDOR	WHAT CHANGED	WHEN
Cursor	Pro moved from 500 fast responses to $20 of usage at API rates; the CEO apologized publicly within three weeks, citing models that "spend more tokens per request on longer-horizon tasks." [5] Teams pricing rebuilt around two usage pools per seat — "a small number of power users on any team tend to drive the majority of spend." [6]	Jun 2025 · Jun 2026
GitHub Copilot	Premium requests with model multipliers, then **AI Credits on every plan, billed per token at each model's published rate**, with no fallback to a cheaper model. "A quick chat question and a multi-hour autonomous coding session can cost the user the same amount." [7]	Jun 2025 · Jun 2026
Anthropic	Weekly limits on Claude Code for individual plans. For Enterprise, a $20 seat with all usage billed at API rates and no bundled tokens. [8]	Aug 2025 · Feb 2026
OpenAI	Codex-only seats with no seat fee, billed on token consumption; credit prices vary roughly 25-fold across models. [9]	Apr 2026

Agentic work — a model planning, editing, running tests and iterating for an hour — consumes something closer to a cloud workload than a software license, and it is now billed like one. **Organizations inherited a variable bill without the instruments variable bills require.** Cloud spend went through this a decade ago and built a discipline

around tags, allocation and unit economics. Coding-agent spend has arrived at cloud-like variability with seat-like visibility: one invoice per vendor, and one admin console per tool showing that tool alone.

What follows is predictable. In May 2026 Forbes, relaying *The Information*, reported that Uber's CTO had confirmed the company exhausted its entire 2026 AI budget by April as Claude Code spread across its engineers, with about 5,000 engineers averaging $150–250 a month and power users at $500–2,000.[3] REPORTED The same month, Microsoft was reported to be cancelling most of its direct Claude Code licenses and moving engineers to GitHub Copilot CLI; the company cited consolidation, and unnamed sources said cost was also a factor.[10] REPORTED In June, Gartner forecast that by 2028 AI coding costs will exceed the average developer's salary: "token discipline will not emerge through developer choice alone."[4]

Spend is not the only thing that moved. Anthropic's own published estimate of what Claude Code costs a typical developer roughly doubled when it was updated in April 2026 — from about $6 to about $13 per active day, and from 90% of users under $12 a day to 90% under $30 — with, in Anthropic's words, "no pricing or product change."[11] The previous figure dated from February 2025; Anthropic attributed the change to Opus 4.7 becoming the primary model and to how usage had changed since.

That is the shape of the problem. The bill is variable, it is growing faster than headcount, and what drives it is behavior, not contract terms. Procurement cannot negotiate it away, because procurement does not decide how long a developer lets an agent run.

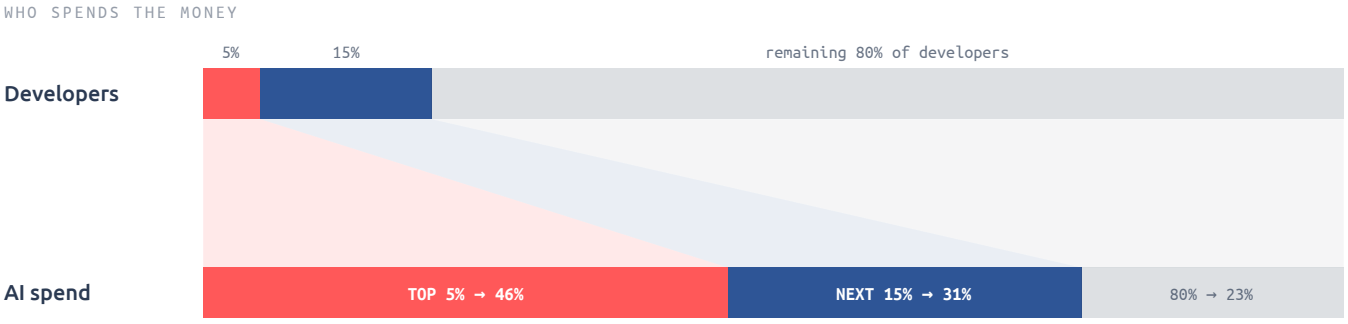
SECTION 03

What makes agent spend different

A seat license has three properties budgets depend on: it is fixed, it is roughly equal across users, and it changes only when someone signs something. Agent spend has none of them.

It is a power law, not an average

Across its user base in spring 2026, Cursor measured a Gini coefficient for AI spend of **0.75**: the top 5% of users account for about 46% of spend, the top 20% for about 77%. [6] Jellyfish's customer telemetry for May 2026 puts the typical user at the median company at $81 a month; across all users, the mean was $248 and the 99th percentile **$2,452 — 35 times the median**. [12] Across firms, Ramp finds about 80% of OpenAI and Anthropic spend comes from 1% of companies. [13]



Source: Cursor, Developer Habits Report, spring 2026 (Gini coefficient 0.75).

An average is almost useless here. "We spend $180 per developer per month" describes almost nobody. It hides a handful of people for whom the agent is effectively a second engineer, and a majority for whom it is a better autocomplete — two investments with different returns that need managing differently.

OBSERVED NIRMATA ENGINEERING · ONE WEEK

37% of the team's output score came from the single most active engineer

79% came from the five most active engineers

The same shape shows up in our own organization, measured on output rather than spend: AIControls' PR-derived productivity score over one recent week. Concentration is not a property of large companies. It appears on a small team in a single week.

Cheaper tokens, larger bills

The price of a token at a given capability is falling fast — by about 5x to 10x a year for frontier models, and far faster on some benchmarks. [14] That is often offered as a reason not to worry. It is not, for three reasons.

Tokens per task are rising faster than price per token is falling. Gartner estimates agentic models need 5–30 times more tokens per task than a chat interaction, and that inference savings "will not be fully passed on." [15] Anthropic's documentation puts an agent-team session at about seven times the tokens of a standard one when teammates run in plan mode. [11]

Context is the cost. Cursor reports that the ratio of input to output tokens in its agents rose from 4.5x to between 11x and 13x between January and May 2026, and that cache reads are now about 90% of all tokens processed. [6] The model re-reads your codebase far more than it writes to it. Spend scales with how much context the agent decides it needs, not with how much code it produces.

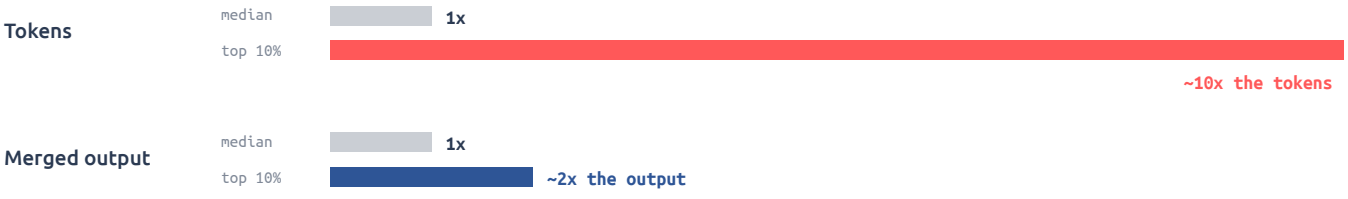
The bill moves when nothing in your organization changes. Jellyfish measured that moving the same workload from one model version to the next raised cost about 40% at an unchanged per-token price, because the new tokenizer counted the same text as more tokens. [12] Anthropic alone changed plan billing or limits at least twice in 2026. [8] Your unit cost can move 40% in a week without a single decision on your side.

The marginal token buys less than the first

Spend and output are not proportional. Across roughly 12,000 developers, Jellyfish found the top decile by token consumption used about ten times the tokens of the median developer for about twice the merged output. [16]

REPORTED

TOP DECILE BY TOKENS, RELATIVE TO THE MEDIAN DEVELOPER



Source: Jellyfish token-usage research, ~12,000 developers (reported).

This is the reason this paper is not an argument for cutting spend. Concentrated in the right places, on the right tasks, by people who use agents well, AI spend returns many times its cost. The same spend in the wrong places returns very little — and nothing in a seat count, a token total or an invoice tells the two apart.

SECTION 04

Where existing measures stop

Most organizations already measure something about their AI coding rollout. None of these measures is wrong, and each is a useful input. Each one answers a different question from the one the budget holder is asking: what did we get for the money?

Adoption and seats

WHAT IT DOES WELL

Shows who has access and who uses it. Adoption is a precondition for return, and a team that never opens the tool will never get anything from it.

WHERE IT STOPS

It measures exposure, not return. An organization with 95% adoption and flat delivery has bought a very expensive autocomplete. One with 40% adoption, concentrated in the teams doing the work agents are good at, may be getting far more.

Lines of code

WHAT IT DOES WELL

Cheap to collect, available everywhere, and a reasonable proxy for effort in a world where every line was typed by a person.

WHERE IT STOPS

AI makes it more misleading than any other measure, because generating lines is exactly what AI is good at. GitClear's analysis of 211 million changed lines found duplicated blocks up eightfold in 2024 and refactored code down about 40%;[17] its 2026 follow-up put refactoring at under 4% of changes, from 21% in 2022.[18] Any ROI figure denominated in lines — including "capacity multipliers" that convert lines into equivalent headcount — will report returns a delivery manager does not recognize.

What developers say

WHAT IT DOES WELL

Captures experience, friction and trust, which matter for adoption and retention and which no telemetry sees. More than 80% of respondents to Google's 2025 DORA survey said AI made them more productive.[21]

WHERE IT STOPS

Developers are poor judges of their own AI-assisted speed. In METR's 2025 randomized trial, experienced open-source developers took 19% longer with AI tools while believing they had been about 20% faster.[19] METR's February 2026 follow-up measured an 18% speedup for returning participants, but called its data "only very weak evidence" for the size of the change, because a growing share of developers would not do half their work without AI.[20] What survives both studies is the perception gap.

Raw pull-request counts

WHAT IT DOES WELL

Counts units of delivered change rather than effort or text, which is why this paper builds on them. In Microsoft's rollout of CLI agents to tens of thousands of engineers, adopters merged about 24% more PRs per engineer than they otherwise would have.[24]

WHERE IT STOPS

A count with no guard rewards exactly the behavior that turns throughput into backlog. Faros AI found high-adoption teams merged 98% more PRs but spent 91% more time in review on PRs 154% larger;[22] a year later, deployments per week had fallen 11.7%, median review time had risen fivefold and incidents per PR by 242%.[23] The Microsoft study did not measure quality or cost, and acknowledges it may miss quality costs.

Vendor consoles and the invoice

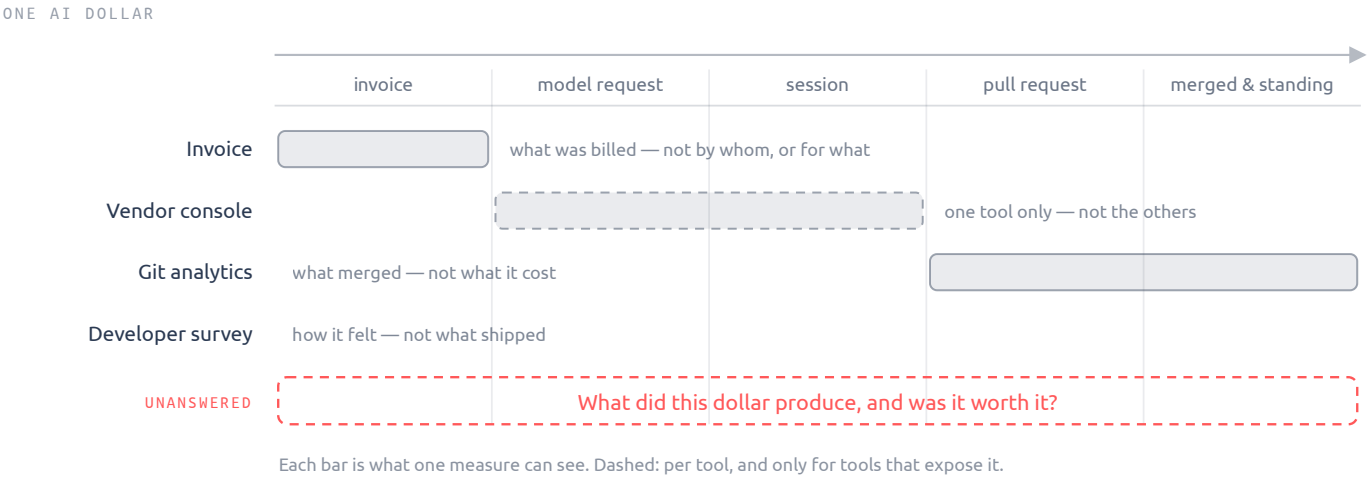
WHAT IT DOES WELL

Authoritative for what was billed and how one tool was used. Since June 2026 some consoles also set budgets.

WHERE IT STOPS

Each sees its own tool, and most organizations run two to four. Finance sees one invoice per vendor and no repository. The FinOps Foundation's guidance makes the limit explicit: tools that bundle usage into the seat can only be measured through what the vendor exposes, and for tools billed to your own provider account it recommends a proxy you control for attribution beyond what the provider exposes.[25]

The pattern across all of them



Four partial views of one dollar. Cost lives on the left of the timeline and outcomes on the right, and no measure most organizations run spans both. ROI needs the cost and the outcome in the same row, attributed to the same people and the same work.

SECTION 05

The unit: cost per accepted outcome

$$\text{AI cost per accepted outcome} = \frac{\text{attributed AI spend in a period}}{\text{accepted outcomes in the same period}}$$

Attributed AI spend is all model and agent spend that can be tied to a developer, to a non-human identity acting for a team, or to a repository — across every tool. It includes seat fees, apportioned; metered usage at the rate actually paid; and spend on failed and abandoned sessions. The FinOps Foundation is explicit: "Failed and discarded runs are cost, not completions." [25] Unattributed spend is reported as its own line, never spread silently across teams. Driving it toward zero is the first operational objective (section 06).

An accepted outcome, for most teams, is a pull request that merged to a protected branch, was not reverted or more than half rewritten within a churn window — we use 14 days — and was not authored entirely by a bot without human review. Teams whose unit of delivery is something else (a deployed change, a closed ticket of a given class) can substitute it, as long as it carries the same survival test.

Guards

Cost per accepted outcome can be gamed exactly as raw PR counts can: split the work and the ratio improves. It is never reported alone. It is read against four guards, each of which should be flat or improving while the ratio improves.

GUARD	WHY IT IS THERE	EVIDENCE
Median PR size	Catches both inflation (giant AI PRs) and gaming (artificially split PRs)	Git
Review time per PR	The cost AI moves onto senior engineers — Faros measured +91%, then fivefold	Code host
Change-failure rate	DORA associates higher AI adoption with lower delivery stability in both 2024 and 2025 [21]	Incidents, rollbacks
Test share of changes	A leading indicator of whether AI output is being verified	Git, session data

Token-only and fully loaded

There are two versions of the number, and both are useful. **Token-only** cost per accepted outcome — AI spend over outcomes — is the one you manage week to week; it moves with model choice, routing, context discipline and coaching. **Fully loaded** cost per accepted outcome — AI spend plus the loaded cost of the engineers involved, over outcomes — answers the CFO's question, because it puts AI spend beside the far larger cost it is supposed to make more productive.

The obvious objection

The strongest argument against everything in this paper: AI spend is small next to salaries, so why manage it at all? Jellyfish has reported AI spend at roughly 2.5% of the marginal cost of a merged change, and fully loaded cost per

merged PR falling about a quarter as adoption rose.[26] REPORTED If that is right, buy more tokens and stop measuring.

It is half right. In the organizations we work with, the fully loaded number usually is dominated by salary, and AI usually does improve it. Four things are still wrong with stopping there:

- **The average hides the tail.** At 35 times the median, the 99th-percentile developer is not spending 2.5% of their loaded cost on AI. The budget overrun in section 02 did not come from the median.
- **The trend is not the level.** A share of 2.5%, when cost per user grew sixteenfold in ten months,[12] will not be 2.5% next year. Gartner's 2028 forecast is the same arithmetic carried forward.
- **The ratio is the opportunity, not the cost.** Some developers get twice the output for ten times the tokens; others get more for less. Closing that gap is worth far more than any token saving, and you cannot close a gap you cannot see.
- **"Cost per PR fell" is not "AI caused it to fall."** Section 07 covers what a defensible claim needs.

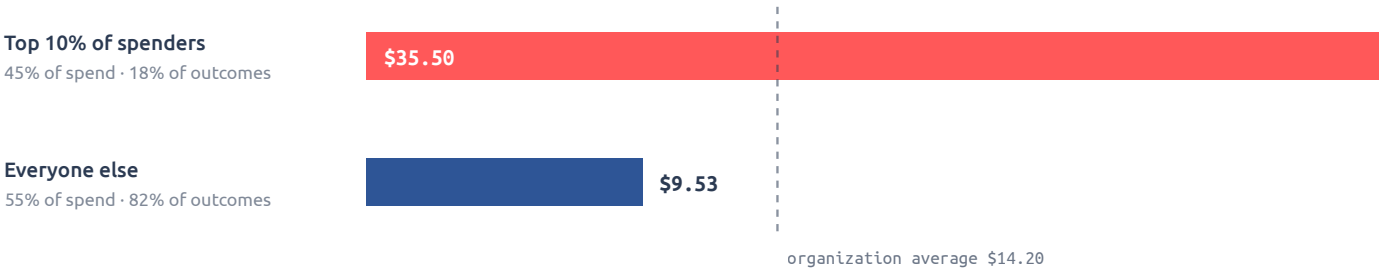
A worked example ILLUSTRATIVE

A 200-engineer organization spends $50,000 a month across three coding tools and merges 4,000 PRs a month. Twelve percent are reverted or substantially rewritten within 14 days, leaving **3,520 accepted outcomes**. Loaded engineering cost is $200,000 per engineer per year — about $3.33 million a month.

$14.20	$961	~1.5%
token-only cost per accepted outcome	fully loaded cost per accepted outcome	AI as a share of fully loaded cost
$50,000 ÷ 3,520	$3.38M ÷ 3,520	so far, the objection holds

Now split the same organization by spend. Suppose the top 10% of engineers by AI spend account for 45% of it ($22,500) and for 18% of accepted outcomes (634).

TOKEN-ONLY COST PER ACCEPTED OUTCOME · ILLUSTRATIVE



The top tenth produces at almost four times the unit cost of everyone else. That may be entirely justified — the hardest problems in the oldest code, where Stanford's study of about 100,000 developers found AI gains smallest.[27] Or it may be agents left running against a context window full of irrelevant files. The organization-level average told you neither. The split tells you where to look, and the answer is usually a coaching conversation, not a budget cut.

OBSERVED THE SAME ARITHMETIC ON OUR OWN ENGINEERING

We run AIControls on Nirmata's engineering organization, which builds AIControls and uses coding agents heavily. One recent week:

477
merged pull requests
GitHub, via AIControls

~$29
token-only AI cost per merged PR
30-day spend, pro-rated

13%
of merged PRs not classified by work type
62 of 477

WORK TYPE	SHARE OF OUTPUT SCORE
Bug fix	45.5%
Feature development	40.8%
Reliability & security	3.7%
Unclassified	3.3%
Infrastructure change	2.3%
Refactoring	1.9%
Testing	1.7%
Documentation	0.5%

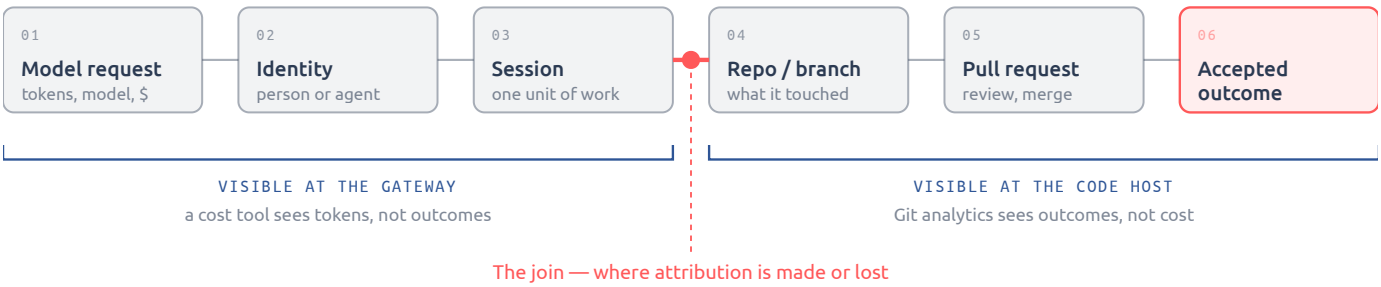
Three things are worth reading from it. The per-PR figure lands close to the $29 per merged PR one other engineering team has published from its own token spend, [30] though both are single organizations. Refactoring is under 2% of output, the pattern GitClear describes across the industry. And the number is not yet a cost per *accepted* outcome: it is not churn-adjusted, and AIControls marks every day of the week as partial evidence. It is the starting point the method in this section improves on, not the finished measure.

SECTION 06

Attribution: from token to pull request

Every number in section 05 depends on joining two datasets that were never designed to meet: what the models were asked to do, and what landed in the repository. The join is the hard part, and it is where most measurement efforts stall.

THE ATTRIBUTION CHAIN



Neither source is enough alone. Links 1–3 can only be established where the request is visible — at a gateway that model traffic passes through. Links 4–6 can only be established from the code host. The FinOps Foundation points the same way: for tools billed to your own provider account, it recommends a proxy for attribution. [25]

LINK	WHAT HAS TO BE KNOWN	WHERE IT USUALLY BREAKS
Request → identity	Which person, or which non-human identity acting for which team, made this call	Shared API keys; agents under a service account with no owner
Identity → session	Which calls belong to one unit of work	Tools that do not expose session boundaries; long-running agents
Session → repo / branch	Which code the session was working in	Calls made outside a git working tree; monorepos
Branch → PR → outcome	Whether the work merged, survived, and what it cost to review	Squash merges; bot-authored PRs; work split across PRs

Unattributed spend is the first number to drive down

When an organization first routes coding-agent traffic through a gateway, the most common finding is not a cost figure. It is how much of the cost cannot be tied to any repository at all. When we first turned AIControls on in Nirmata's own engineering organization, before repository detection was configured, 97% of sessions and 89% of spend appeared as *unattributed* — developers working outside detected repositories, agents under shared credentials, tools whose sessions carried no working-directory context. OBSERVED Those are onboarding figures, not a steady state: they describe what a gateway sees on day one, and they fall as repository detection, identity mapping and agent ownership are configured.

Report unattributed spend as its own line, never pro-rated across teams, and treat reducing it as the first milestone of the rollout. An ROI figure computed on the attributed 30% of spend is not an ROI figure.

Multi-tool estates

Most organizations now run several coding agents at once. Attribution has to be tool-agnostic, or "AI cost per PR" means something different on every team. Routing every tool's model traffic through one gateway is the simplest way to get a single unit. Where a tool cannot be routed — some IDE-bundled assistants do not allow it — import its usage export, attribute at the coarsest level it supports, and label it as such.

Non-human identities

Background agents that open PRs, triage issues or fix failing builds are a growing share of both spend and output. In Nirmata's own tenant a third of the identities active in a month were not people. OBSERVED Each needs an owner accountable for its spend, and its outcomes measured on the same terms as a person's. An agent that opens fifty PRs a week, of which three survive, has a cost per accepted outcome that is very easy to compute and very rarely computed.

PER-DEVELOPER DATA IS FOR COACHING, NOT RANKING

Attribution to individual developers is necessary — without it, the most important finding in section 03 cannot be acted on. It is also the fastest way to lose the trust of the people being measured. The organizations that get this right make three commitments, in writing, before the data exists:

1. Individual cost-per-outcome figures are visible to the developer and their manager, not published.
2. They are used to find coaching opportunities, not to rank people. Jellyfish attributes about three-quarters of the variance in AI-assisted output to the individual engineer and almost none to model choice [16] — an argument for teaching, not for league tables.
3. No compensation or performance decision is made on them alone.

A metric becomes a target the moment it is ranked, and cost per PR is easy to game by splitting work. The guards in section 05 catch gaming at team level. Only these commitments prevent it at individual level.

SECTION 07

Baselines, ROI and benchmarking

Cost per accepted outcome tells you what you are paying. ROI needs a second number: what you would have produced without the spend. That counterfactual is never observed directly, so every ROI claim rests on a baseline — and the choice of baseline decides the answer more than anything else.

BASELINE	WHAT IT IS	STRENGTH	WEAKNESS
Your own history	Same teams, same repositories, before or with less AI use	Controls for codebase, process and people	Pre-AI data may be stale; teams change
Open-source cohorts	Comparable public repositories, with and without heavy AI use	Large samples; transparent	Different review culture, incentives and scope
Published research	Figures from controlled studies and industry reports	Methodology published; well known	Built for other codebases; often LOC-based; ages quickly

The most defensible is usually your own history, adjusted for headcount and scoped to the same repositories. Use the other two as cross-checks, not substitutes.

Why line-count multipliers mislead

It is easy to build an ROI figure that looks extraordinary. Take a published lines-per-developer-per-day estimate from before AI, compare it with the lines your AI-assisted team added last week, and report the ratio as a "capacity multiplier" converted into equivalent headcount. Against a baseline of a few hundred lines a day, a small team using agents heavily will routinely show twentyfold multiples and "hundreds of FTEs" of added capacity.

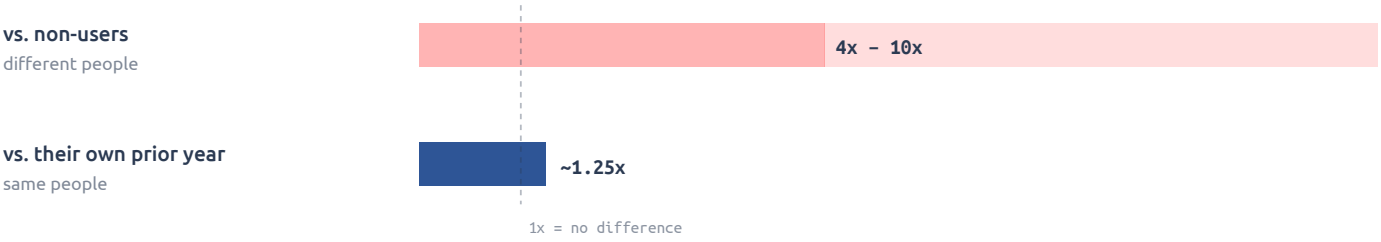
We ran it on ourselves. Against a baseline of 280 lines per developer per day, the capacity view in AIControls reports a **20x multiplier** for Nirmata's engineering team in one recent week — "equivalent to 307 additional full-time developers," or 19.2 full-time developers added for every $1,000 a month of AI spend. OBSERVED The product labels it an extreme estimate, and it is right to. The same week produced 477 merged pull requests, which is a great deal of output for a small team and nothing like the output of three hundred engineers.

No engineering leader should put that number in front of a board, and none we talk to believes it. It is not false so much as meaningless: it measures text, and the business did not buy text. The fix is to change both sides of the ratio — accepted outcomes in the numerator, a baseline of the same unit from the same kind of work in the denominator. Where size matters, weight outcomes by a capped size measure or a complexity class, not by raw lines.

Association is not causation

Even with the right unit, a before-and-after comparison shows association. Teams adopt AI at the same time as they change process, hire, lose people, take on easier or harder roadmaps, and adopt other tools.

SAME DATA, TWO COMPARISONS — OUTPUT OF HEAVY AI USERS



Most of the raw gap was selection, not AI. GitClear's 2026 cohort study: the developers who use AI most were already the most productive. [28] For regular users, the remaining ~25% is still a real and valuable effect — it is just not a tenfold one.

A defensible ROI claim therefore compares like with like (same teams, same repositories, adjusted headcount); reports a range, not a point — DORA's ROI guidance says the same; [29] states the coverage of its evidence (how much spend was attributed, how many PRs had resolvable authors, which days had partial data); and says plainly, on the report, that it shows association. That is a higher standard than most vendor ROI claims meet. It is also the standard a CFO will apply.

Benchmarking against other organizations

Your own history tells you whether you are improving. It does not tell you whether you are good. Peer benchmarks — organizations of similar size, stack and AI maturity, measured the same way — answer questions internal data cannot. Is our cost per accepted outcome high because our work is hard, or because our agents are badly configured? Is a 35x spread between our median and 99th-percentile developer normal? How much of peer spend goes to frontier models, and does it show up in outcomes?

Benchmarks are only as good as their consistency. A peer figure is useful only when computed with the same attribution, the same outcome definition and the same churn window. Otherwise it is a different number with the same name.

HOLD CONSTANT	WHY A PEER FIGURE FAILS WITHOUT IT
Attribution	An organization at onboarding, with most spend unattributed, and one with 5% unattributed are not reporting the same cost
Outcome definition	Merged PRs, accepted PRs and deployed changes differ by multiples, not percentages
Churn window	A 14-day window and no window produce different denominators from the same work
Cost basis	List price, negotiated price and credit-funded usage give different numerators
Size band	A 15-person team and a 1,500-person organization do not share a distribution

This paper publishes no cross-organization averages. A peer figure is evidence only once the cohort behind it is large enough to report without identifying anyone in it, and computed on the terms above.

SECTION 08

Levers that move cost per outcome

Once cost per accepted outcome is visible, it can be managed. The levers below are ordered roughly by impact against effort. Not every one suits every organization.

01 Model routing

Model choice is the largest single driver of token cost and, by most measures, one of the smallest drivers of output. Jellyfish puts it under 1% of the variance in merged output per engineer.[16] Cursor's own benchmark shows its in-house model completing tasks at about $0.55 against $11.02 for a frontier model at maximum effort, for a 1.6-point difference in score — twenty times the cost for a marginal gain.[6] Routing sends each request to the cheapest model that can do it — by task class, repository, developer preference within a policy, or automatic fallback. The guard: watch PR survival on cheaper routes.

02 Budgets and alerts

The FinOps Foundation's guidance: budgets at 110–120% of a 30- to 60-day observed baseline, alerts at 80% and 100%, moving from showback to chargeback as attribution matures.[25] Budgets belong where someone can act on them — team and individual, not just organization. A budget inside one vendor's console covers one tool; a budget that covers one tool in three is a budget for a third of the problem.

03 Plan capacity before metered spend

Most organizations pay for both subscription plans with usage windows and metered API access. Routing that exhausts included plan capacity first, then overflows to pooled provider credits or metered access, reduces spend without reducing usage. Startups holding provider credits can stack them the same way.

04 Context and session hygiene

Input tokens dominate cost, so what an agent reads matters more than what it writes. Scoping sessions to a task, ending long-idle sessions, keeping repository instructions concise and avoiding sessions that re-read the same large files repeatedly cut cost per session without cutting output. Session-level visibility — duration, turns, repeat reads, cache hit rate — is what makes it possible.

05 Coaching

The individual accounts for most of the variance in AI-assisted output,[16] so the highest-leverage intervention is teaching. Developers who make heavy use of reusable agent skills show about 30% higher throughput;[16] Microsoft found the strongest predictor of adopting agents was whether peers already had.[24] Find the developers with high outcomes at low cost, learn what they do differently — task selection, prompting, session scoping, review discipline — and spread it. The developers with high cost and low outcomes are not a problem to be capped. They are the audience.

06

Review load

AI moves work from writing to reviewing. If review time per PR is rising, every other lever is partly illusory: the cost has moved onto senior engineers' calendars, where token reports cannot see it. Smaller PRs, tests written with the code, AI-assisted first-pass review and clear ownership of agent-authored PRs are the familiar responses. The point is to measure review load beside cost, so the trade is visible.

LEVER	TYPICAL EFFECT	EFFORT	MAIN GUARD
Model routing	High, on token cost	Low	PR survival on cheaper routes
Budgets and alerts	Caps the tail	Low	Starving critical work
Plan and credit pooling	Medium, on token cost	Low	—
Context and session hygiene	Medium, on token cost	Medium	Output per session
Coaching	Highest, on outcomes	Medium–high	Trust (section 06)
Review load	Protects outcomes	Medium	Change-failure rate

SECTION 09

Can you answer these about your own organization?

Eight questions. There is no score and no maturity level — the point is to find the ones you cannot answer, because those are the gaps, and which ones they are matters more than how many.

The *What we usually find* line under each is OBSERVED rather than published: drawn from the organizations we work with, not from a survey, and offered as a starting expectation to check against your own.

01 What was your AI cost per accepted PR last month, by team?

ASK OF YOUR ENVIRONMENT

Not total AI spend, and not PRs merged. Both, in the same row, for each team, for the same month.

IF YOU ARE EVALUATING A VENDOR

Ask them to produce it from your data during the evaluation — across every coding tool you use, not only theirs.

WHAT WE USUALLY FIND

Spend is known by vendor. PR counts are known by repository. Nobody has joined them.

02 What share of your AI spend cannot be attributed to any repository?

ASK OF YOUR ENVIRONMENT

After everything you can tie to a developer and a codebase, what is left?

IF YOU ARE EVALUATING A VENDOR

Ask whether unattributed spend is shown as its own line or quietly spread across teams.

WHAT WE USUALLY FIND

On day one, most of it.

03 Who are your top 5% of spenders, and what share of accepted outcomes do they produce?

ASK OF YOUR ENVIRONMENT

If they account for nearly half of spend, do they account for anything like half of the output?

IF YOU ARE EVALUATING A VENDOR

Ask to see spend and outcomes plotted per developer on the same chart.

WHAT WE USUALLY FIND

A handful of developers whose agents produce extraordinary value, and a handful whose agents mostly read files.

04 If a vendor changed its pricing or limits tomorrow, how long until you knew what it cost you?

ASK OF YOUR ENVIRONMENT

GitHub, Anthropic, Cursor and OpenAI each changed pricing or limits at least once in 2026.

IF YOU ARE EVALUATING A VENDOR

Ask how a change in a provider's per-token price or tokenizer shows up in their reports, and how quickly.

WHAT WE USUALLY FIND

At the next invoice.

05 What baseline does your AI ROI figure use, and would your CFO accept it?

ASK OF YOUR ENVIRONMENT

Which unit, which period, which teams, and what is excluded?

IF YOU ARE EVALUATING A VENDOR

Ask what their headline ROI number is denominated in. If the answer is lines of code, ask for the same figure in accepted PRs.

WHAT WE USUALLY FIND

A vendor-supplied percentage based on acceptance rate or a developer survey.

06 Is review time per PR going up or down since you rolled out agents?

ASK OF YOUR ENVIRONMENT

And is anyone accountable for the answer?

IF YOU ARE EVALUATING A VENDOR

Ask whether they report review load beside cost and output, or only the latter two.

WHAT WE USUALLY FIND

Up — and noticed by senior engineers months before it appears in any report.

07 Which of your active AI identities are not people, and who owns each one's spend?

ASK OF YOUR ENVIRONMENT

Background agents, CI-integrated agents, bots that open PRs.

IF YOU ARE EVALUATING A VENDOR

Ask for an inventory of non-human identities derived from traffic, not from configuration.

WHAT WE USUALLY FIND

The agents registered through a platform team are known. The ones set up with a personal API key are not.

08 Could you cut AI spend 30% next quarter without cutting output — and do you know where?

ASK OF YOUR ENVIRONMENT

Which models, which sessions, which teams?

IF YOU ARE EVALUATING A VENDOR

Ask them to name the three largest savings in your data, with the expected effect on outcomes.

WHAT WE USUALLY FIND

Yes, mostly through routing and session hygiene — but nobody can show the effect on outcomes in advance, so the change never gets made.

SECTION 10

One layer, not another dashboard

Everything described here needs three things: a point where every model request is visible and attributable; a connection to the code host, where outcomes are recorded; and a place where the two are joined, read against a baseline, and acted on. Most organizations have the second. Few have the first. Fewer still have the third.



Two vantage points, one join. The gateway sees every request; the code host sees every outcome. Neither replaces the coding agents developers already use.

AIControls, from Nirmata, is built as that layer. It sits in the path of coding-agent traffic as a gateway that works with any model provider and any agent, and it connects to GitHub. From those two vantage points it provides:

- **Budgets and cost attribution**, enforced at the gateway across every tool routed through it — by organization, team, developer, project and session — with unattributed spend shown as its own line.
- **Outcomes**: merged PRs, gateway activity and sessions on one time axis per team and repository, with the coverage of the evidence shown alongside, and the explicit caveat that co-movement shows association.
- **AI ROI against a chosen baseline** — your own history, open-source cohorts or published research — with every assumption visible and editable.
- **Peer benchmarking**: your cost and outcome figures against anonymized organizations of similar size, computed the same way.
- **Model routing, failover and credit pooling** — levers 01 and 03 — applied at the same point where cost is measured.
- **Session records**: duration, turns, work type, cache efficiency, and quality and safety signals such as test share, churn and sensitive-file access — the inputs to coaching.

The same gateway that measures cost is where agent behavior is governed: which models, tools and MCP servers an agent may use, what it may send and receive, and what evidence is kept. Nirmata's companion paper, *Governing AI agents in Kubernetes*, covers that half. Organizations that start with cost and productivity usually find they have built most of the foundation for governance as well — and the reverse.

None of this requires replacing the coding agents your developers already use. It requires seeing what they cost, what they produce, and how the two relate — in one place, in one unit.

Start with the number you do not have

An AIControls ROI assessment produces your organization's cost per accepted outcome, its attribution gap and its peer comparison — from your own traffic and repositories. For teams of up to 50 developers, a self-serve 15-day trial does the same without a sales conversation.

nirmata.com/aicontrols-platform

About Nirmata

Nirmata is the AI governance company for Kubernetes. Its platform authorizes what AI agents may call, on whose behalf and within what budget, enforcing those decisions at runtime rather than documenting them afterwards. Governance runs on Kyverno, the CNCF-graduated policy engine Nirmata created and contributed to the CNCF in 2020, applied across four control points — pipeline, admission, gateway and runtime — so infrastructure security, compliance and AI governance are enforced by the same rules and recorded in the same audit trail. Products include AIControls, Nirmata Control Hub and Nirmata Enterprise for Kyverno. Kyverno remains Apache 2.0 open source, with no fork and no lock-in.

APPENDIX A

About the Nirmata data

Every figure marked OBSERVED comes from AIControls running in Nirmata's own engineering organization. It is one organization, and an unusual one: it builds AIControls and uses coding agents more heavily than most. Read it as a worked example of the method, not as a benchmark.

ITEM	BASIS
Outcomes	Merged pull requests imported from GitHub over one week in October 2026. Not churn-adjusted. Pull requests with unresolved authors excluded.
Output score	AIControls' PR-derived productivity score, used only for concentration and work-type shares. Work type is assigned by automated assessment or manual choice; 62 of 477 PRs were unclassified.
Spend	Gateway-measured AI spend over the surrounding 30 days, pro-rated to seven days for the per-PR figure.
Attribution	Share of sessions and spend not tied to a detected repository at onboarding, before repository detection was configured. An initial figure, not a current one.
Evidence coverage	AIControls marks all seven days as partial evidence: collected data was retained, but completeness of collection could not be confirmed.
Privacy	No individual is named. Concentration figures report shares only.

APPENDIX B

Sources

1 PUBLISHED **Anthropic, PBC, Form S-1 Registration Statement**, October 2026 — p. 134 and "Select Defined Terms." Net Revenue Retention is Qualified Customers' revenue in the three months ending on the measurement date divided by their revenue in the three months ending a year earlier, excluding Claude Pro and Max subscribers.
sec.gov/Archives/edgar/data/2130357/000162828026067168/anthropicpbc-sx1.htm

2 PUBLISHED **Menlo Ventures, *2025: The State of Generative AI in the Enterprise***, 9 December 2025. Survey of 495 buyers.
menlovc.com/perspective/2025-the-state-of-generative-AI-in-the-enterprise/

3 REPORTED **Forbes, "Uber Burns Its 2026 AI Budget In Four Months On Claude Code"**, 17 May 2026, relaying reporting by *The Information*. Budget amount not disclosed.
forbes.com/sites/janakirammsv/2026/05/17/uber-burns-its-2026-ai-budget-in-four-months-on-claude-code/

4 PUBLISHED **Gartner, "Gartner Predicts AI Coding Costs Will Surpass Average Developer Salary by 2028 as Token Consumption Surges"**, press release, 24 June 2026.
gartner.com/en/newsroom/press-releases/2026-06-24-gartner-predicts-ai-coding-costs-will-surpass-average-developer-salary-by-2028-as-token-consumption-surges

5 PUBLISHED **TechCrunch, "Cursor apologizes for unclear pricing changes that upset users"**, 7 July 2025, quoting Cursor's CEO.
techcrunch.com/2025/07/07/cursor-apologizes-for-unclear-pricing-changes-that-upset-users/

6 PUBLISHED **Cursor, Teams pricing announcement (June 2026) and *Developer Habits Report* (spring 2026, data to 16 May 2026)**.
cursor.com/blog/teams-pricing-june-2026 · cursor.com/insights/developer-habits

7 PUBLISHED **GitHub, "GitHub Copilot is moving to usage-based billing"**, 27 April 2026, effective 1 June 2026; and GitHub changelog, 18 June 2025 (premium requests).
github.blog/news-insights/company-news/github-copilot-is-moving-to-usage-based-billing/

8 PUBLISHED **Claude Code limits and Enterprise billing**: TechCrunch, 28 July 2025 (weekly limits); The Register, 16 April 2026 (Enterprise seat with usage at API rates); BleepingComputer, 29 August 2026 (weekly limit change).
theregister.com/software/2026/04/16/anthropic-ejects-bundled-tokens-from-enterprise-seat-deal/5226555

9 PUBLISHED **OpenAI, "Codex flexible pricing for teams"**, 2 April 2026; Codex rate card (help.openai.com/en/articles/20001106).
openai.com/index/codex-flexible-pricing-for-teams/

10 REPORTED **Fortune, "Microsoft's AI cost problem"**, 22 May 2026, citing The Verge.
fortune.com/2026/05/22/microsoft-ai-cost-problem-tokens-agents/

11 PUBLISHED **Anthropic, Claude Code documentation, "Manage costs"**, accessed October 2026; change in estimate reported by Business Insider, 29 April 2026.
code.claude.com/docs/en/costs

12 PUBLISHED **Jellyfish, *State of AI in Software Engineering*, H1 2026**.
jellyfish.co/state-of-ai-software-engineering/

13 PUBLISHED **Ramp, AI Index**, June and October 2026.
ramp.com/data/ai-index-oct-2026

14 PUBLISHED **Epoch AI, "LLM inference price trends"**, 12 March 2025; MIT FutureTech, arXiv:2511.23455, November 2025.
epoch.ai/data-insights/llm-inference-price-trends

15 PUBLISHED **Gartner, press release on LLM inference cost**, 25 March 2026.
gartner.com/en/newsroom/press-releases/2026-03-25-gartner-predicts-that-by-2030-performing-inference-on-an-llm-with-1-trillion-parameters-will-cost-genai-providers-over-90-percent-less-than-in-2025

16 REPORTED **Jellyfish token-usage research as presented at AI Engineer**, reported by WorkOS (5 August 2026) and StartupHub (22 October 2026).
workos.com/blog/nicholas-arcolano-jellyfish-token-maxing-diminishing-returns · startuphub.ai/ai-news/artificial-intelligence/2026/your-model-isn-t-the-bottleneck-jellyfish-finds

17 PUBLISHED **GitClear, *AI Copilot Code Quality***, February 2025; summarized by DevClass, 20 February 2025.
devclass.com/2025/02/20/ai-is-eroding-code-quality-states-new-in-depth-report/

18 PUBLISHED **GitClear, *The AI Code Quality Maintainability Gap***, 2026.
gitclear.com/the_ai_code_quality_maintainability_gap

19 PUBLISHED **METR, "Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity"**, 10 July 2025.
metr.org/blog/2025-07-10-early-2025-ai-experienced-os-dev-study/

20 PUBLISHED **METR, uplift study update**, 24 February 2026.
metr.org/blog/2026-02-24-uplift-update/

21 PUBLISHED **Google Cloud DORA, *Accelerate State of DevOps* 2024 and *State of AI-assisted Software Development* 2025**.
cloud.google.com/blog/products/ai-machine-learning/announcing-the-2025-dora-report

22 PUBLISHED **Faros AI, *The AI Productivity Paradox***, July 2025.
faros.ai/blog/ai-software-engineering

23 PUBLISHED **Faros AI, *AI Acceleration Whiplash***, Q2 2026.
faros.ai/research/ai-acceleration-whiplash

24 PUBLISHED **Murphy-Hill, Butler, Savelieva et al. (Microsoft), arXiv:2607.01418**, 1 July 2026.
arxiv.org/abs/2607.01418

25 PUBLISHED **FinOps Foundation, *Tokenomics***, last updated 3 June 2026; *State of FinOps 2026*.
finops.org/wg/token-economics-saas/ · data.finops.org

26 REPORTED **Jellyfish conference talk, 13 October 2026**, as summarized by Adyog Pulse.
pulse.adyog.com/insights/jellyfish-cost-per-merged-pr-fell-26-but-it-doesn-t-capture-review-eff

27 REPORTED **Y. Denisov-Blanch (Stanford), "Does AI actually boost developer productivity?"**, AI Engineer talk, 2025. Not peer-reviewed.
ai.engineer/talks/tbDDYKRFjhk-ai-developer-productivity

28 PUBLISHED **GitClear, *Developer Cohort Analysis: AI Coding Output***, 4 January 2026.
gitclear-public.s3.us-west-2.amazonaws.com/Developer_Cohort_Analysis_AI_Coding_Output.pdf

29 PUBLISHED **DORA, *ROI of AI-Assisted Software Development***, 2026.
dora.dev/ai/roi/report/

30 PUBLISHED **CampusIQ, "What we actually spend"**, 16 October 2026. $53,000 of token spend over 1,832 merged PRs, four engineers; tokens only, not fully loaded.
campusiq.com/blogs/what-we-actually-spend

Model and product names are as each vendor printed them on the date cited. Pricing and plan limits in this market change frequently; every figure is dated, and none should be read as current beyond its date.

Nirmata · October 2026 · Comments and corrections on this document are welcome.