



APRIL 2026

The Rework Tax

Understanding the real cost
of rebuilds and **rollbacks**

Why AI coding bills ran past annual budgets in early 2026, how much of that spend paid for code that did not last, and what the evidence says about engineering process as the cause.

Five findings

-
- 01 Budgets broke early.** Uber's CTO said the company spent its planned 2026 AI coding budget in about four months.¹ In 2025, Anthropic added weekly limits after one \$200-a-month subscriber consumed "tens of thousands" of dollars in model usage.²
-
- 02 Spend grows much faster than output.** The heaviest token users in Jellyfish's Q1 2026 sample shipped about twice the merged pull requests of the lightest, at \$89.32 per PR against \$0.28.⁷
-
- 03 Much accepted code is later rewritten.** Waydev reports initial acceptance of 80–90% falling to 10–30% once later revisions are counted.⁶ Faros AI measured an 861% rise in code churn under high AI adoption.⁸
-
- 04 Review is the new bottleneck.** In the same Faros data, median time in review rose 441.5%, PRs merged with no review rose 31.3%, and incidents per merged PR rose 242.7%.⁸
-
- 05 The link to architecture is plausible but not proven.** No source here measures design scrutiny against token spend directly. Usage incentives, consumption pricing and always-on agents are separately documented drivers of the same bills.
-

SCOPE AND METHOD

All sources were published on or before 30 April 2026 and are numbered on page 10. Most quantitative findings come from engineering-analytics and cost-governance vendors, each with a commercial interest in the problem; their methods differ, so figures are reported side by side and never combined. *Rework* here means code that is revised, replaced or deleted soon after it is merged. Where the report gives an inference rather than a sourced fact, it says so.

The bills

The clearest public case through April 2026 is Uber. In an interview with The Information, chief technology officer Praveen Neppalli Naga said the company had used up its planned 2026 AI coding budget within about four months.¹

32% → 84% Claude Code adoption among ~5,000 engineers, February to March	\$150–250 Average monthly spend per engineer; heavy users near \$2,000	\$1,200 Reportedly spent by the CTO in a single two-hour demo
--	--	---

Figures as reported by The Information.¹ Uber also ran internal leaderboards that ranked engineers by tool activity.

Vendors saw the same pattern from the other side. On 28 July 2025, Anthropic announced weekly rate limits for its Pro and Max plans, saying some users were running Claude Code "continuously in the background, 24/7." It estimated the limits would affect fewer than 5% of subscribers, and said they also targeted account sharing and resale.^{2,3} A month earlier, Cursor had moved its Pro plan toward usage-based pricing, then apologised and offered refunds to users surprised by charges.⁴

Survey data suggests the forecasting problem is widespread. A 2025 study of 372 enterprises found only 15% forecast AI costs within 10% of actual, and nearly one in four missed by more than 50%.⁵ Its authors sell cost-governance software.

"One user consumed tens of thousands in model usage on a \$200 plan."

ANTHROPIC, 28 JULY 2025²

Where the tokens go

A token bill records how much work the model did, not whether that work survived. Engineering-analytics firms began measuring the gap in 2025 and early 2026, and their results point the same way: AI tools produce more code, and a larger share of it is later changed or deleted.

10–30%

of AI-generated code still in place after follow-up revisions, against 80–90% initial acceptance

WAYDEV, VIA TECHCRUNCH, 17 APRIL 2026⁶

Waydev works with 50 customers employing more than 10,000 engineers. Its CEO, Alex Circei, told TechCrunch that managers see high acceptance rates but miss the churn in the weeks that follow.⁶ GitClear's January 2026 cohort analysis found regular AI users averaged 9.4 times higher code churn than non-users, a larger multiple than the productivity gain it measured.⁹

Faros AI's 2026 report, drawn from two years of telemetry on 22,000 developers across more than 4,000 teams, found code churn (lines deleted relative to lines added in merged code) rose 861% between each organisation's lowest and highest periods of AI adoption. Work restarts rose 13.8%, and 26% more in-progress tasks sat idle for a week or longer.⁸

More duplication

GitClear's 2025 study of 211 million changed lines found an eightfold rise during 2024 in code blocks with five or more duplicated lines.¹⁰

Less consolidation

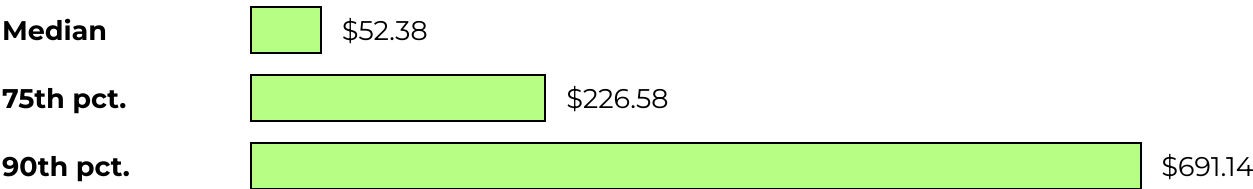
Moved lines, its proxy for refactoring, fell 39.9%. 2024 was the first year copy-pasted lines outnumbered moved lines.¹⁰

Our reading: duplication and reduced consolidation are signs of code added without reference to the structure already in place.

Diminishing returns

If rework is part of the bill, heavy users should pay more per unit of output. Jellyfish's analysis of 12,000 developers at 200 companies priced Q1 2026 usage at published Claude API rates and joined it to merged pull requests. Weekly merged PRs rose from 0.77 at the lowest usage to 2.15 at the highest, while tokens per PR rose from about 7 million at the median to 69 million in the top decile.⁷

FIGURE 1 · ESTIMATED MONTHLY TOKEN SPEND PER DEVELOPER



Q1 2026, priced at published Claude API rates.⁷

FIGURE 2 · LIGHTEST VS HEAVIEST TOKEN USERS, FULL QUARTER

Spend Tier	Token Spend	Merged PRs	Cost per PR
Bottom 20%	\$3	11	\$0.28
Top 20%	\$1,822	23	\$89.32

About 600 times the spend for about twice the merged PRs.⁷

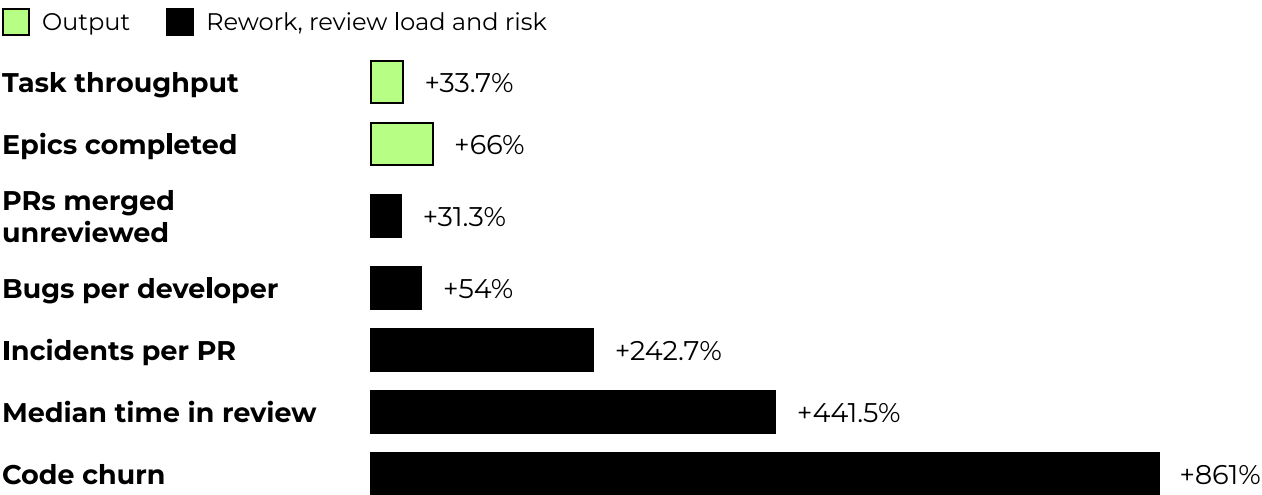
Jellyfish does not attribute the gap to rework. It points to an "agentic barrier": autonomous agents need sandboxed environments, orchestration and context engineering before they pay off, and more tokens do not get an organisation past that limit.⁷ That explanation is still architectural. It places the cost in the systems around the model, not in the model's price.

The scrutiny gap

The thesis depends on one mechanism: code enters the system before anyone checks whether it fits, and is paid for again when it has to be fixed, replaced or removed. No public data measures design review directly. Code review is measured, and the trend there is consistent.

Under high AI adoption, Faros found output rising alongside review load, defects and incidents. Bugs per developer rose 54%, up from 9% in its 2025 report. Faros attributes the rise in unreviewed merges to reviewers who "cannot keep pace," not to a policy decision.⁸

FIGURE 3 • CHANGE FROM LOWEST TO HIGHEST AI ADOPTION



Shared linear scale. 22,000 developers across 4,000+ teams, each organisation compared with itself.⁸

Output rose by tens of percent. Review time, incidents and churn rose by hundreds.

Almost right

Developer surveys describe the same failure at the level of individual work. Code that is almost right is the kind most likely to pass a quick review and be rebuilt later.

84% use or plan to use AI tools	66% cite "solutions that are almost right, but not quite"	45% say debugging AI-generated code takes more time
---	---	---

Stack Overflow 2025 Developer Survey, about 49,000 respondents.¹³

Faster feels faster

In a July 2025 randomised trial, METR found 16 experienced open-source developers took 19% longer with AI tools on their own repositories. They expected a speed-up and still believed afterwards they had been faster.¹⁴ The study is small and predates 2026's agentic tools.

Bigger changes, less stability

Google's 2024 DORA report associated AI adoption with slightly better code quality but an estimated 7.2% drop in delivery stability, noting that the larger changes AI makes easy are slower and more prone to instability.¹¹

Our reading: the gap between perceived and measured speed is the same gap that makes token spend hard to judge from inside a team. Reviewers and architects absorb the instability first. When they cannot keep up, more of it reaches production, and removing it is paid for in tokens as well as hours.

Testing the thesis

The thesis: runaway token use is a consequence of low architectural and process scrutiny, with tokens spent building things that later have to be rebuilt or removed. Four bodies of evidence complicate it.

Incentives explain some spend without any rework

Uber ranked engineers by tool activity.¹ When usage itself is rewarded, spend can rise regardless of what the code needs. TechCrunch described large token budgets as "a badge of honor" among developers.⁶

Pricing changes moved the bill, not the work

Cursor's shift toward usage pricing and Anthropic's weekly caps changed what users paid for similar workloads.^{3,4} Anthropic tied its outliers to continuous background use and account resale, not rework.²

Churn is not always waste

Faros offers three explanations for its 861% figure: rework of AI code that proved insufficient, large refactors AI made affordable, and engineers improving code they were never satisfied with. It says all three fit the data and the mix varies by organisation.⁸

Sources disagree on whether good foundations protect you

DORA's 2025 report concludes from survey data that strong engineering foundations amplify AI's benefits and buffer its downsides.¹² Faros's telemetry found mature organisations showing the same downstream deterioration as everyone else.⁸

CAVEAT Who produced the numbers

Waydev, GitClear, Faros and Jellyfish sell engineering analytics; Mavvrik sells cost governance. Each has a commercial interest in showing AI spend needs measuring. Their methods differ: Faros compares each organisation with itself over time, Jellyfish compares developers within one quarter, GitClear compares cohorts. The direction of their findings agrees. The magnitudes should not be combined or averaged.

What the evidence supports

ESTABLISHED

Large organisations exceeded AI coding budgets well ahead of plan. Token cost per unit of output rises steeply at high usage. Code produced under heavy AI use is revised or deleted at several times the prior rate. Review queues have lengthened, and more code merges without review.

INFERRED

A meaningful share of high-end token spend pays for code that is later rebuilt, and some of that rework comes from work that started without enough design or review. This follows from combining the churn, review and cost data. No single dataset ties a given dollar of spend to a given deleted line.

UNKNOWN

What fraction of runaway spend is rework rather than incentive-driven or always-on usage. Whether teams with formal design review before implementation see lower churn and lower cost per merged change. No public study through April 2026 measures this directly.

Read together, the sources describe a rework tax: a cost paid in tokens, review time and incidents for code that was generated quickly and then had to be revisited. They show the tax exists and grows with usage. They do not yet show how much of any company's bill it accounts for, or how much design scrutiny upstream would reduce it.

MEASUREMENTS THAT WOULD SETTLE IT

- 1 Line-level provenance showing whether deleted code was recently generated. Faros notes this can already separate rework from legacy refactoring.⁸
- 2 Token spend attributed to the changes it produced, not to the engineer or the month.
- 3 A record of which work passed design review before implementation, to compare churn and cost with and without it.

Numbered references

All published on or before 30 April 2026. Vendor research is marked **V**. Figures attributed to The Information are as reported by that publication.

1. The Information. [Uber CTO shows Claude Code can blow AI budgets](#), Apr 2026.
2. Anthropic. [Weekly rate limits for Claude Pro and Max, X](#), 28 Jul 2025.
3. TechCrunch. [Anthropic unveils new rate limits to curb Claude Code power users](#), 28 Jul 2025.
4. Cursor. [Clarifying our pricing](#), 4 Jul 2025.
5. **V** Mavvrik and Benchmarkit. [State of AI Cost Governance 2025](#), 372 enterprises.
6. Tim Fernholz, TechCrunch. ['Tokenmaxxing' is making developers less productive than they think](#), 17 Apr 2026.
7. **V** Nicholas Arcolano, Jellyfish. [Is "tokenmaxxing" cost effective?](#), 15 Apr 2026.
8. **V** Faros Research. [AI Engineering Report 2026: The Acceleration Whiplash](#), 12 Apr 2026.
9. **V** GitClear. [Developer Cohort Analysis: AI Coding Output](#), Jan 2026.
10. **V** GitClear. [AI Copilot Code Quality: 2025 Research; summary by DevClass](#), 20 Feb 2025.
11. Google Cloud. [Announcing the 2024 DORA report](#), Oct 2024.
12. DORA. [State of AI-Assisted Software Development 2025](#).
13. Stack Overflow. [2025 Developer Survey: AI](#), Jul 2025; coverage in [The Register](#), 29 Jul 2025.
14. METR. [Measuring the impact of early-2025 AI on experienced open-source developer productivity](#), 10 Jul 2025.

Several sources here measure different things in different ways. Read each figure against its own method before comparing it with another.