

The AI and Software Engineering *Report 2026*

Everything the research actually
says. In one place.



OCTOBER 2026

The Main Findings

We read every serious study we could find on what AI has done to software engineering - the productivity trials, the code quality analyses, the security testing, the payroll data. We put them all here, and for each one we've said who ran it, who paid for it, and whether they had anything to gain.

55% faster

In a controlled trial, developers using AI finished a coding task in 71 minutes instead of 161. The study ran with 95 freelance developers building an HTTP server in JavaScript, and less experienced coders gained the most.

WHAT THE RESEARCH FOUND

Faster



55%

Slower



19%

AI-generated code with security flaws



45%

19% slower

A different trial took experienced developers working on codebases they knew well. They were 19% slower with AI.

They believed they'd been 20% faster. None of them noticed.

45% security flaw rate

Share of AI-generated code that contains a security vulnerability, tested across more than 100 models and 80 coding tasks.

Newer and larger models were no better.

CODE QUALITY

**Refactoring
down 70%.**

**Duplication
up 81%.**

Across 623 million code changes, every maintainability signal moved the wrong way at once.

2024 was the first year on record where copy/paste exceeded refactored code.

HIRING

**Juniors
down 20%.**

**Seniors
up 6-12%.**

Payroll data covering millions of US workers. Developers aged 22–25 declined nearly 20% since late 2022.

Workers 30 and over in the same jobs grew.

84% use it. 29% trust it.

Adoption is close to universal. Trust is not.
The engineers closest to production trust it least.

The short version

We read every serious study on AI and software engineering we could find and put them in one place. Here's what they say.

AI made writing code faster and judging it harder.

The speed gains are real and well evidenced. Boilerplate, tests, scaffolding, the first draft of almost anything — genuinely quicker, and the gains are biggest for less experienced developers.

But the same research shows what came with it. More code shipped, and more of it breaking. Duplication climbing, refactoring collapsing, security flat. And a gap

between how fast people feel and how fast they are that nobody in the study noticed.

Which changes who you need to hire.

AI can do what a junior does. It writes the boilerplate, the simple fix, the test script. It cannot do what a senior does, because someone now has to catch the 45% that fails a security test and the duplication that's up 81%.

The data backs that up: where AI automates a task, entry-level hiring falls. Where it augments, employment holds. So the market repriced experience. Not our opinion — Stanford's payroll data.

WHAT THAT MEANS FOR YOU

Finding and hiring engineers got harder, not easier. There's more code to review, less signal in a resume, and the people who can tell good from plausible are worth more than they were three years ago.

01 | THE PARADOX

Two Studies, *Opposite Answers*

In 2022, GitHub and Microsoft ran a controlled experiment. Ninety-five freelance developers were asked to build an HTTP server in JavaScript. Half got Copilot, half didn't. The Copilot group finished in 71 minutes. The control group took 161.

Fifty-five percent faster. Statistically significant. The result that launched a thousand pitch decks.

In 2025, a research group called METR ran a different experiment. They took sixteen experienced developers working on large open-source codebases they'd built themselves and knew intimately. Real tasks — actual bugs and features from their own backlogs. They randomized which ones got AI and which didn't.

Before starting, the developers predicted AI would save them about a quarter of their time. Afterwards, they estimated it had saved about 20%.

They were 19% slower.

Neither study is wrong. They measured different things, and the difference is the entire point of this report.

“

AI is fast at writing code from scratch and slower at changing code that already exists.”

What actually changed

The GitHub task was greenfield. A new HTTP server, from nothing, in a language with enormous training data behind it. That's the best case for AI and it's a genuinely common kind of work.

The METR tasks were the opposite. Mature code, deep context, conventions the developer had spent years building. The slowdown was worst where they had the most prior experience with the codebase.

So the honest summary is narrower than either headline: AI is fast at writing code from scratch and slower at changing code that already exists. Most professional engineering is the second thing.

Where the time went

This is the part worth sitting with, because it explains everything downstream.

AI removes the parts of coding that feel like work. The typing. Looking up the argument order. The scaffolding you've written a hundred times. Those disappear, and the session feels lighter.

But the time moved rather than vanished. It went into writing prompts, reading what came back, and working out whether it's actually right. In the METR study, more than half of AI suggestions were unusable, and the ones developers kept still needed fixing by hand.

You stop writing and start reviewing. Reviewing is slower, and it doesn't feel slower — which is why not one of sixteen experienced developers noticed they'd lost three hours a day.

The same pattern, at scale

Google's DORA team surveyed nearly 5,000 technology professionals and found the same shape in the aggregate data.

AI adoption now correlates with higher throughput — teams do ship more. It also correlates with higher delivery instability, and that hasn't improved.

They tested whether AI's speed compensates: if you're shipping faster, can you fail fast and fix fast? The answer was no. The instability isn't offset by the velocity.

Their conclusion is the cleanest sentence anyone has written about this:

AI amplifies whatever is already there.

Teams with strong review, testing and platform practices get faster. Teams without them get worse faster.

What follows from this

If more code is being produced, and less of it is being written from first principles, and the people producing it can't reliably tell how well it's going — then the bottleneck moved.

It used to be writing. It's now judgment. Review, architecture, knowing what good looks like, catching the thing that compiles and runs and is quietly wrong.

That's not a junior skill. And it turns out the hiring market noticed before most companies did.

→ *The rest of this report is the evidence, study by study, including who paid for each one.*

The Studies

STUDY 01

55% faster on a task built from scratch



AT A GLANCE

2022 | 95 Developers | JavaScript Task | Randomized Trial

GitHub and Microsoft — Peng et al., 2023

What they did

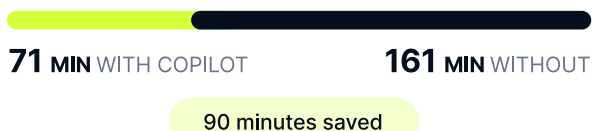
Recruited 95 freelance developers and asked them to build an HTTP server in JavaScript. Randomly assigned half to use GitHub Copilot and half to work without it. Measured time to completion.

What they found

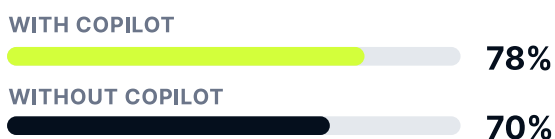
55.8% faster

statistically significant at $p=.0017$, with a 95% confidence interval running from 21% to 89%.

AVERAGE COMPLETION TIME

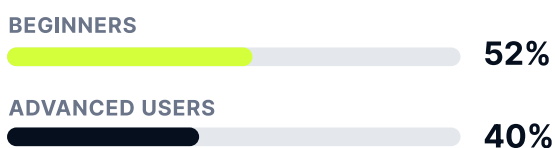


TASK SUCCESS



Task success also improved, though less dramatically

SEPARATE ANZ BANK TRIAL



42% faster overall. The gains were largest for less experienced developers.

Who ran it, and what they sell

GitHub, who make Copilot, in partnership with Microsoft, who own GitHub. This is the vendor measuring its own product. Read 55% as a best case.

It's worth saying the study is methodologically sound — randomized, controlled, properly reported, with the confidence interval published rather than hidden. The issue isn't rigor. It's what was measured.

What it doesn't tell you

The task was greenfield. A new HTTP server, from nothing, in JavaScript — a language with enormous training data behind it, solving a problem that has been solved in public thousands of times.

That is the single most favorable case for an AI coding tool, and it is not what most professional engineering looks like.

The participants were also freelancers recruited through Upwork. About 45% were self-employed, roughly half earned under \$10,000 a year, and around 5% were unemployed. A Google study that ran a similar experiment with its own engineers, in a codebase they knew, specifically noted this as a limit on how far the original result generalizes.

What it means if you're hiring

Don't dismiss this number. AI speeds up boilerplate, scaffolding, tests and first drafts. Engineers should use AI for these tasks.

This does not mean you need fewer engineers. The faster work was never the hard part.

STUDY 02



19% slower, while feeling 20% faster

AT A GLANCE

2025 | 16 Developers | Open-source code | Randomized trial

METR — randomized controlled trial, 2025

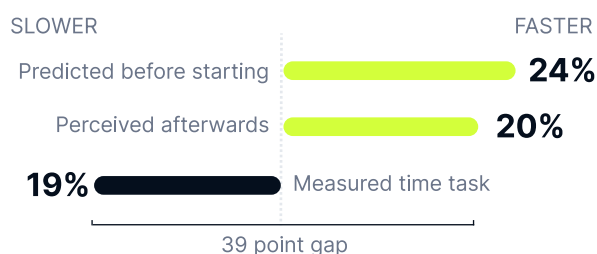
What they did

Recruited 16 experienced open-source developers, averaging around five years of work on the repositories in question. Took 246 real tasks from their own backlogs — genuine bug fixes and features they needed to do anyway. Randomized which tasks could be done with AI tools and which couldn't. Timed everything.

What they found

THE 39 POINT GAP BETWEEN FEELING AND FACT

What developers expected AI to do for their speed, what they thought happened afterward, and what was actually measured.



That's a gap of roughly 39 points between perception and measurement. Not one participant noticed it while it was happening. More than half of AI suggestions were discarded, and the ones developers kept still required manual correction.

Who ran it, and what they sell

METR is an independent research nonprofit. They don't sell coding tools and have no commercial interest in the result. They also published the finding despite it contradicting their own prior expectations, which is the behavior you want from a research group.

What it doesn't tell you

Sixteen developers is a small sample. The design is careful, but it is not a large study.

The population is specific and it matters: experienced engineers working in codebases they built and know deeply. The slowdown was worst where prior familiarity was highest. On unfamiliar code, new projects and boilerplate, other research consistently finds real speedups.

So this is not evidence that AI makes developers slower. It's evidence that AI makes these developers slower on *this* kind of work.

The tools were also early-2025 vintage. A follow-up found the original cohort still around 18% slower, but newly recruited developers only about 4% slower — so the gap may be closing as tooling and habits mature.

What it means if you're hiring

The measurement problem is the finding, not the 19%.

If sixteen experienced engineers can lose three hours a day and report the opposite, then self-reported productivity is not evidence of anything.

Neither is a candidate telling you how much AI has improved their output.

Watch what someone does with the tool. Don't ask them how it's going.

STUDY 03



45%

of AI-generated code has a security flaw

AT A GLANCE



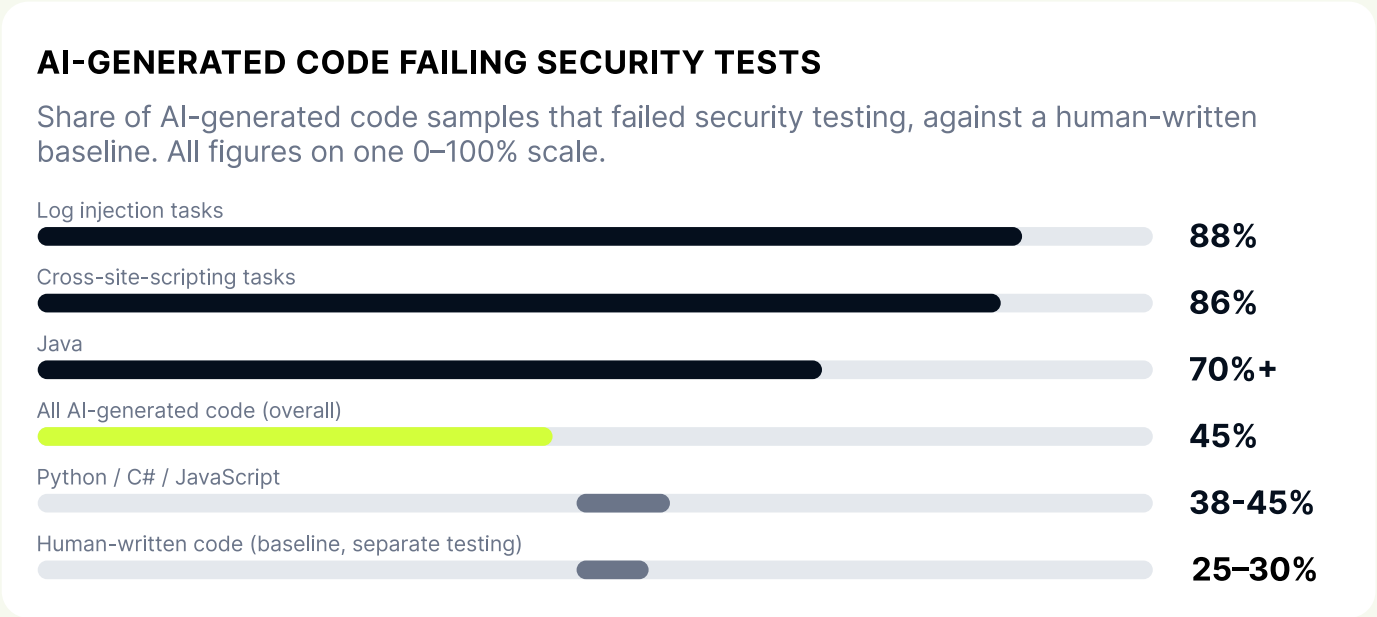
Veracode — 2025 GenAI Code Security Report

What they did

Gave 80 curated coding tasks to more than 100 large language models, across Java, Python, C# and JavaScript.

Ran every output through security testing against four OWASP categories: SQL injection, cross-site scripting, log injection, and insecure cryptography.

What they found



The finding that should bother you most: security performance was flat across model generations. Larger models weren't safer than smaller ones. A March 2026 update found the pass rate essentially unchanged at around 55% — over a period when coding benchmarks like HumanEval improved steadily. Models got better at writing code and no better at writing safe code.

Who ran it, and what they sell

Veracode sells application security testing. Their business depends on code having flaws, so read 45% as an upper bound rather than a central estimate. That said, the methodology is published, the sample is large, and the vulnerability classes are specific — this isn't a vendor blog asserting a number.

What it doesn't tell you

It tested raw model output, not code that's been through a review. Separate testing of agentic coding tools found zero exploitable SQL injection or XSS, which suggests tooling and workflow matter more than the model. Human-written code isn't clean either — comparable testing puts it around 25–30%.

What it means if you're hiring

Somebody has to catch this. That somebody needs to know what an authorization flaw looks like when it's buried in code that runs perfectly and passes every test you wrote.

Authorization logic was the persistent weak point across every tool tested, because it varies by application and needs reasoning about the whole system. That's the definition of senior work.

STUDY 04



More throughput, more instability

AT A GLANCE

2025 | 5,000 Professionals | Global survey | 100+ Interview hours

Google DORA — State of AI-assisted Software Development, 2025

What they did

Surveyed nearly 5,000 technology professionals worldwide and conducted over 100 hours of qualitative interviews. Used Bayesian statistical modeling with explicit causal assumptions, controlling for individual experience, available resources and organizational stability.

What they found

AI adoption has reached roughly 90% of developers. It now correlates positively with software delivery throughput — a reversal of DORA's 2024 finding, when AI correlated with *reduced* throughput. Teams have adapted, and they ship more.

Delivery instability also continues to rise. More change failures, more rework, longer recovery times.

The researchers specifically tested whether AI's speed compensates for this — the "fail fast, fix fast" hypothesis. It doesn't. The instability isn't offset by the velocity.

AI showed no measurable effect on either friction or burnout. Those turn out to be properties of the organization, not the workstation.

Who ran it, and what they sell

Google, who sell AI coding tools and cloud infrastructure. The obvious incentive would be to report that AI helps unambiguously — and they didn't. DORA has published findings unflattering to AI two years running, including the 2024 result that adoption reduced throughput.

Methodologically this is the strongest survey-based work in the field.

What it doesn't tell you

It's correlational. Bayesian modeling with stated causal assumptions is better than raw correlation, but it isn't a controlled trial. Teams that adopt AI heavily may differ from teams that don't in ways the controls don't capture.

What it means if you're hiring

DORA's own summary is the most useful sentence in this report: AI amplifies whatever is already there. Good teams get faster. Struggling teams get worse, faster.

That has a direct hiring implication. Adding AI to a team without strong review practices increases output and increases breakage simultaneously. The constraint isn't how much code you can generate — it's how much you can competently review.

STUDY 05

623 million code changes.

Every maintainability signal moved the wrong way.



AT A GLANCE

2023–2026 | 623 Million code changes | 6 Quality signals | Repository analysis

GitClear — The Maintainability Gap, June 2026

What they did

Analyzed 623 million code changes between 2023 and 2026, drawn from anonymized commercial repositories and major open-source projects. Classified each change by operation — added,

deleted, moved, copy/pasted, churned — and tracked six distinct quality signals over the period. Roughly a quarter of commits in the dataset show measurable AI assistance.

This is not a survey. Nobody was asked their opinion.

What they found

EVERY REPORTED SIGNAL WORSENE

⬇️ Refactoring

-70%

against a 2022 baseline

⬆️ Copy/paste within commits

9.4% → 15.7%

2022 to H1 2026

⬇️ Function connectivity

-35%

new code calling existing code

⬆️ Code block duplication rose 81%

+81%

40.3 → 73.0 blocks / M lines. The highest on record.

⬆️ Error-masking constructs

+47%

constructs that hide failures

⬆️ Two-week code churn

+15%

code changed again within two weeks

Their earlier report found 2024 was the first year on record where copy/paste exceeded moved code. Moved code is the signal of refactoring, and it had been the larger of the two for the entire history of the dataset.

Who ran it, and what they sell

GitClear sells code review and analytics tools, so they benefit commercially from concern about code quality.

The counterweight is that this is the largest dataset in the field by a wide margin, the classification scheme is unglamorous and published, and the trends run consistently across multiple independent reports over four years. It's harder to dismiss than a survey.

What it doesn't tell you

It's correlational. AI adoption rose over the same period as these trends, and this data cannot prove one caused the other. Team sizes, hiring patterns, remote work, and post-2022 market conditions all changed

at the same time. It also measures signals that *proxy* for maintainability rather than maintainability itself. Duplication isn't always bad. Sometimes copying is the right call.

What it means if you're hiring

Duplication is the cheapest thing in the world to create and the most expensive thing to fix, because every copy has to be found before any of them can be corrected.

The skill that's now scarce is the one that stops this happening — noticing that the thing being written already exists somewhere else, and caring enough to go and find it. That is experience, and it doesn't show up in a take-home test.

STUDY 06



Juniors down 20%. Seniors up 6-12%.

AT A GLANCE

2021-2025 | ADP Payroll data | Millions of workers | Occupational analysis

Stanford Digital Economy Lab — ADP payroll data, 2025

What they did

Used payroll records from ADP, the largest payroll processor in the United States, covering millions of workers across thousands of firms. Tracked employment in AI-exposed occupations from 2021 through mid-2025, broken down by age.

Payroll records, not job postings and not a survey. This is what employers actually paid people.

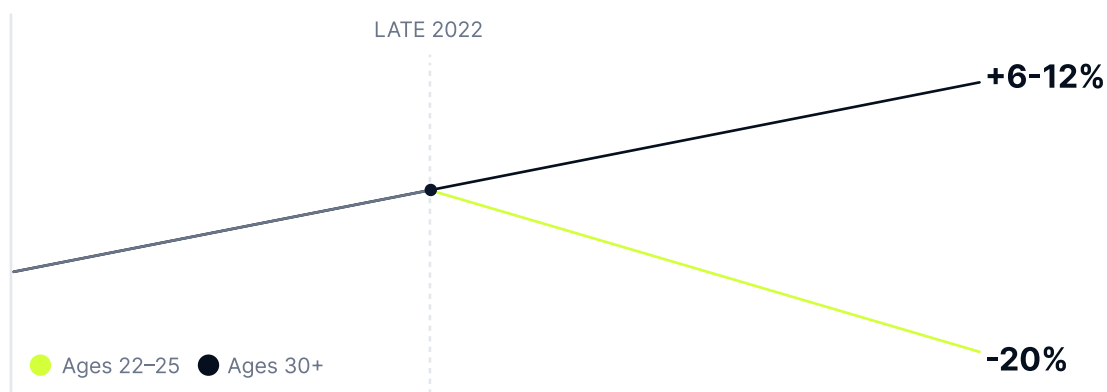
What they found

Software developers aged 22 to 25 declined nearly 20% from a peak in late 2022. Workers aged 30 and over in the same high-exposure occupations grew by 6% to 12% over the same period.

Before late 2022 the two groups tracked together. The divergence begins almost exactly when generative AI tools went mainstream.

TWO GROUPS, ONE TURNING POINT

Employment in AI-exposed software roles, by age group, 2021 through mid-2025. Illustrative trend based on the report's reported before/after figures — exact indexed values by month aren't published.



Before late 2022, both groups tracked together. They diverged as generative AI went mainstream.

The mechanism is the interesting part. Where AI automates a task — writing basic functions, simple fixes, test scripts — entry-level hiring falls. Where AI augments a task — supporting problem-solving, checking work — employment holds steady or grows. Software development landed in the first category.

The adjustment happened through reduced hiring rather than pay cuts or layoffs.

Who ran it, and what they sell

Stanford's Digital Economy Lab, led by Erik Brynjolfsson. Academic research with no product attached and no commercial interest in the finding.

What it doesn't tell you

Timing is the real problem here. The end of zero-interest-rate policy compressed entry-level tech hiring independently of AI, and the two effects overlap almost exactly. The researchers address this by showing results hold separately for occupations with high and low interest-rate exposure, but the confound can't be fully removed.

A Danish study using similar methodology found near-zero employment effects. That's a serious challenge to the result and it deserves to be stated plainly.

Reported figures also vary between 13%, 16% and 20% depending on the cutoff and comparison group used. The direction is consistent; the magnitude isn't settled.

What it means if you're hiring

The market repriced experience before most companies noticed.

If you're hiring engineers, the pool of experienced people didn't grow — demand for them did, while the entry point into the profession narrowed. That makes senior hiring more competitive, not less.

And it raises a question the industry hasn't answered: if nobody hires juniors now, where do the seniors come from in ten years? We don't have a good answer to that either.

THE LONG-TERM RISK

The pipeline has a gap where the entry point used to be.

Senior hiring is becoming more competitive while the entry point into the profession narrows.

If nobody hires juniors now, where do the seniors come from in ten years?



STUDY 07



84% use It. 29% trust It.

AT A GLANCE

2025 | Global developers | AI tool adoption | Self-reported survey

Stack Overflow Developer Survey

What they found

Adoption is close to universal — 84% of developers use or plan to use AI tools, up from 76% the previous year.

Trust went the other way. The share who trust the accuracy of AI output fell from 40% to 29%. Active distrust sits at 46%, meaning more developers distrust AI output than trust it. Only about 3% report high trust.

The texture of the complaint is specific: 66% say the core problem is solutions that are almost right but not quite, and 45% say debugging AI-generated code takes longer than writing it themselves would have.

And the correlation that matters most — skepticism rises with experience. Experienced developers show the lowest high-trust rate of any group at 2.6%, and the highest strong-distrust rate at 20%. The people closest to production code trust it least.

Who ran it, and what they sell

Stack Overflow, whose traffic and business model have been damaged by AI assistants answering the questions developers used to ask them. That's a real incentive to

report skepticism, and it should be weighed.

Against that: the survey is long-running, the sample is very large, and the methodology has been consistent for years.

What it doesn't tell you

It's entirely self-reported, which makes it the weakest evidence in this report. Given METR found developers badly wrong about their own productivity, there's no obvious reason to think they're well calibrated about trust either.

Stack Overflow's respondents also skew toward a particular kind of developer — people who participate in developer communities — which may not represent the whole profession.

What it means if you're hiring

Treat this one as a directional signal rather than a measurement.

The signal worth having is the experience correlation. If the developers with the most production exposure are the most skeptical, then a candidate with uncritical enthusiasm for AI output is telling you something about how much of their code has been on call at 3am.

02 | THE STATE OF HIRING

The Signal *Collapsed*

Everything so far is about what AI did to code. This section is about what it did to the process of finding the person who writes it.

Applications roughly doubled. Then they doubled again.

in

11,000

job applications per minute.

45% increase year on year

+45.5%

applications

-10.6%

job postings

LinkedIn now processes around 11,000 job applications per minute, a 45% increase year on year, and LinkedIn's own research finds US applicants per open role have doubled since spring 2022. In Q3 2024 the platform recorded a 45.5% surge in applications at the same time as a 10.6% drop in postings — more people chasing fewer jobs, applying faster.

The mechanism isn't mysterious. 74% of US job seekers now use AI somewhere in their application process. Paste the posting into a chatbot, get a tailored resume and cover letter, send it, repeat.

Recruiters interviewed by WIRED in 2026 described candidate pools going from around 100 applicants per role to over 1,000 in less than eighteen months. The response has been to deliberately add friction back into application forms, reversing a decade of one-click-apply. One recruiter's summary:

“*we kind of want friction now.*”

The resume stopped being evidence.

This is the part that matters more than the volume.

A resume used to be a weak signal. It's now close to no signal, because the thing that used to differentiate a good one — clear writing, relevant framing, tailoring to the role — is exactly what a language

model produces in four seconds. Take-home tests went the same way, for the same reason.

So the two cheapest filters in hiring both stopped working at once, precisely when the volume going through them multiplied.

CLEAR WRITING

→

**AI-GENERATED
IN SECTIONS**

→

WEAKER SIGNAL

TAKE-HOME TESTS

→

**AI-ASSISTED
OUTPUT**

→

WEAKER SIGNAL

Time to hire went up, not down.

Here the data is genuinely messy and we're not going to pretend otherwise. SHRM's benchmarking puts average US time to hire at around 44 days in 2026, up from roughly 36 in 2019. Industry benchmarks for technology and software sit at 30 to 42 days. Some specialist sources report 60 to 95 days for senior engineering roles, and one puts Bay Area senior hiring at 67 days in Q1 2026 against 38 the previous quarter.

Those numbers don't reconcile, and anyone quoting a single one as fact is being careless. What they agree on is direction: hiring engineers takes longer than it did, seniority adds roughly 20% at every level, and the pressure is upward.

The reason is straightforward. More applications, worse signal per application, so more interview rounds to compensate.

One benchmark found companies running 42% more interviews per hire than in 2021.

And what companies need changed.

AI-related job postings are growing significantly faster than overall hiring, and the number of distinct AI job titles on LinkedIn expanded from 860 in 2022 to over 3,000 by early 2026.

Meanwhile the entry point narrowed. Stanford's payroll data, covered earlier in this report, shows developers aged 22–25 down nearly 20% while workers over 30 in the same occupations grew.

So the demand didn't fall. It moved — toward people who can direct the tools, review the output, and catch the 45% that fails a security test.

Which leaves most companies in the same position.

More applicants than ever. Less able to tell which ones are any good. Competing for a narrower pool of experienced engineers. Running longer processes to compensate. That's the hiring problem AI created. It isn't a shortage of code. It's a shortage of judgment, on both sides of the interview.

What To Actually *Do About It*

We're not going to pretend this page is neutral — we place engineers for a living. But most of what follows you can do without talking to anyone.

STOP DOING THIS



Stop screening on the resume

It's the filter that broke first and hardest. Use it to check for basic relevance and nothing else. Whatever you were reading into how well one was written, stop.



Stop take-home tests as a filter

You're measuring what someone produces with unlimited time and a model, which is neither the job nor a differentiator. Strong candidates won't do them anyway.

START DOING THIS



Watch someone work instead

45 minutes of live coding on a real task tells you more than any artifact they send you. It's the only part of the process AI hasn't degraded, because you're watching the thinking rather than reading the output.



Interview for review

The scarce skill is spotting the thing that compiles, runs, passes the tests, and is wrong. Give a candidate AI-generated code with a flaw and ask what they make of it. Most processes don't test this at all.



Let them use AI in the interview

Banning it tells you how someone codes in conditions that no longer exist. What you're assessing isn't whether they use it — it's whether they read what it produced.



Add friction on purpose

One question that takes real thought to answer will remove more noise from your pipeline than any screening tool.



Move faster once you find someone good.

With senior processes running past 60 days, the constraint isn't finding people. It's that anyone worth hiring has two other offers by week five.

We've written up how we run live coding interviews — the actual internal document our engineers use, including the scoring rubric and the parts we've got wrong. That and a few other things are at

cloudemployee.io/resources

Sources

Peng, Kalliamvakou et al. — *The Impact of AI on Developer Productivity: Evidence from GitHub Copilot*, GitHub / Microsoft, 2023

METR — *Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity*, 2025

Google DORA — *State of AI-assisted Software Development*, 2025

GitClear — *The Maintainability Gap: AI Code Quality in 2026*, June 2026

GitClear — *AI Copilot Code Quality*, February 2025

Veracode — *2025 GenAI Code Security Report*, and March 2026 update

Stanford Digital Economy Lab — *Canaries in the Coal Mine? Six Facts about the Recent Employment Effects of Artificial Intelligence*, 2025

Stack Overflow — *Developer Survey*

Chatterjee et al. — *GitHub Copilot productivity study*, ANZ Bank

LinkedIn application volume data, as reported by The New York Times, 2025

Greenhouse — *2025 AI in Hiring Report*

SHRM — *Recruiting Benchmarking Report*, 2025

WIRED — *reporting on recruiter response to application volume*, 2026

Cloud Employee

Over 300 engineers working inside companies across the US and UK. Operating since 2014.