



2026 · EDITION 01

A POINT OF VIEW

Code is cheap.
Trust isn't.

What changed when AI started writing software, what the evidence says, and what leaders should build next.

Anyone can generate it. Few can prove it.

Sanjeev Azad

October 2026

research cutoff 8 october 2026
evidence marked throughout

— A NOTE FROM THE AUTHOR

The expensive part got *cheap*.

For most of my career, writing code was the expensive part of building software. We hired for it, planned around it and measured it. That assumption broke sometime in the last two years.

Today an agent can read a repository, change forty files and open a pull request before I finish my coffee. Google says three quarters of its new code is now generated by AI and approved by engineers. Nine in ten professional developers use a coding agent every week. Code is no longer scarce.

Something else is. When I talk to engineering leaders, the question has changed from "how do we write this faster?" to "how do we know this is right?" Review queues are longer. Security teams are nervous. Junior engineers are harder to place. The bill for tokens is growing faster than anyone planned.

This book is my attempt to describe that shift honestly. It pulls together more than sixty sources, from randomised trials to earnings calls, and keeps them apart so you can weigh them yourself. Where I give my own view, I say so.

My argument is simple. The next advantage in software engineering will not come from generating more code. It will come from being able to trust what gets generated, and from building the systems, skills and habits that earn that trust.

I hope it helps you decide where to put your next dollar and your next hire.

Sanjeev Azad

October 2026

CONTENTS

Inside this *book*

Executive summary	04
The year in numbers	05

PART I

Code got cheap

01	Five capabilities that now overlap	07
02	The landscape	08
03	From vibe coding to agentic engineering	09
04	How much code AI now writes	10

PART III

New economics, new building blocks

09	Cheaper tokens, bigger bills	17
10	Benchmarks run out of road	18
11	A model built to decide: System One and RLCD	19
12	Where a decision model fits	20

PART V

People and the road ahead

17	The broken first rung	27
18	The engineer's new job	28
19	Three futures, 2027 to 2030	29
20	The first 90 days	30
	The leadership agenda	31

PART II

Trust got harder

05	The assurance gap	12
06	The productivity puzzle	13
07	Where the time goes	14
08	The new attack surface	15

PART IV

Verified delegation

13	The intelligence stack	22
14	Intent to Assurance	23
15	Autonomy follows evidence	24
16	Verification gets cheaper too	25

BACK MATTER

Evidence and terms

Method	32
Sources	33
Glossary	35
About the author	36

How to read the marks

- PRIMARY STUDY
- SURVEY
- VENDOR
- ANALYST
- OUR CONCEPT

Every number carries its evidence class. A company's statement about its own product is not a controlled study. A survey tells you what people say, not what they ship. A forecast is a forecast. I keep them apart on purpose, because the most common mistake in this debate is to stack a vendor headline on a survey and call it a trend.

EXECUTIVE SUMMARY

AI made code cheap. It did not make software *trustworthy*.

The organisations that win the next five years will be the ones that close that gap first. Seven findings explain why.

01 Generation went mainstream in under two years.

Google's AI share of new code went from about a quarter to three quarters in eighteen months. One in three pull requests on GitHub now involves an agent. ● **VENDOR**

02 Assurance did not keep pace. Almost all AI-generated code is now syntactically correct, yet only about 56% passes security checks, and that rate has barely moved. ● **VENDOR** When maintainers reviewed 296 agent pull requests for METR, they would not have merged roughly half of the ones that passed the tests. ● **PRIMARY****03 Productivity gains are real but conditional.** Lab studies show large effects; field studies show small or uncertain ones. In the best-known trial, developers believed they were faster than they were. ● **PRIMARY****04 The bottleneck moved downstream.** More code means bigger pull requests, longer reviews and lower stability, unless testing and platforms are already strong. AI amplifies the team you already have. ● **VENDOR DATA** ● **SURVEY****05 Agents opened a new attack surface.** Prompt injection is structural, not a bug to patch, and attackers already use developers' own agents against them. ● **PRIMARY****06 The economics are inverting.** Tokens get cheaper every year, yet spend per developer is rising, and per-seat pricing has given way to usage billing. ● **VENDOR****07 The first rung of the career ladder is breaking.** Employment of developers aged 22 to 25 is down about 20% from its 2022 peak, and learning with AI can weaken the skills needed to check AI. ● **PRIMARY**

WHAT TO DO ABOUT IT

Treat verification as a product, not a phase. Fund tests, review capacity and security analysis alongside every dollar spent on generation.

Measure cost per accepted change, not lines generated or share of AI code.

Widen what agents may do only where you can check the result, contain the failure and recover from it.

Invest in specifications and evaluation sets, which are now worth more than the code, and protect the junior pipeline that will one day supervise the machines.

THE YEAR IN NUMBERS

Six numbers that tell the *story*.

Six recent figures. The left column is about speed. The right column is about trust.

75%

of new code at Google is AI-generated and approved by engineers

Google · Apr 2026

● VENDOR

56%

of AI code-generation tasks produce secure code, a rate that has barely moved as models improved

Veracode · Jul 2026

● VENDOR

90%

of professional developers use AI coding agents at least weekly; 68% use them every day

JetBrains · Aug 2026, n > 15,000

● SURVEY

~50%

of agent pull requests that passed the tests would not be merged by the project's maintainers

METR · 296 PRs, Mar 2026

● PRIMARY

1 in 3

pull requests on GitHub now involves an AI agent

Microsoft · FY26 Q4 call, Jul 2026

● VENDOR

-20%

nearly 20% fall in employment of software developers aged 22 to 25 since the late-2022 peak

Stanford DEL · ADP payroll data

● PRIMARY

AI doesn't fix a team; it amplifies what's already there.

DORA, State of AI-assisted Software Development 2025

How to read the marks

Every number carries its evidence class. The full key is on the contents page.

● PRIMARY STUDY

● SURVEY

● VENDOR

● OUR CONCEPT

I

PART ONE

Code got *cheap*

In under two years, writing code went from the scarce part of software engineering to the cheap part. This part shows how fast it happened, who builds the tools, and how much of the writing machines now do.

4mo

The time it takes for the length of software task frontier AI can finish, at 50% reliability, to double. It has held roughly that pace since 2023.

● PRIMARY

METR, Time Horizon 1.1, 2026

01

Five capabilities that now overlap

02

The landscape

03

From vibe coding to agentic engineering

04

How much code AI now writes

Code is cheap. Trust isn't.

CHAPTER 01 · FIVE CAPABILITIES THAT NOW OVERLAP

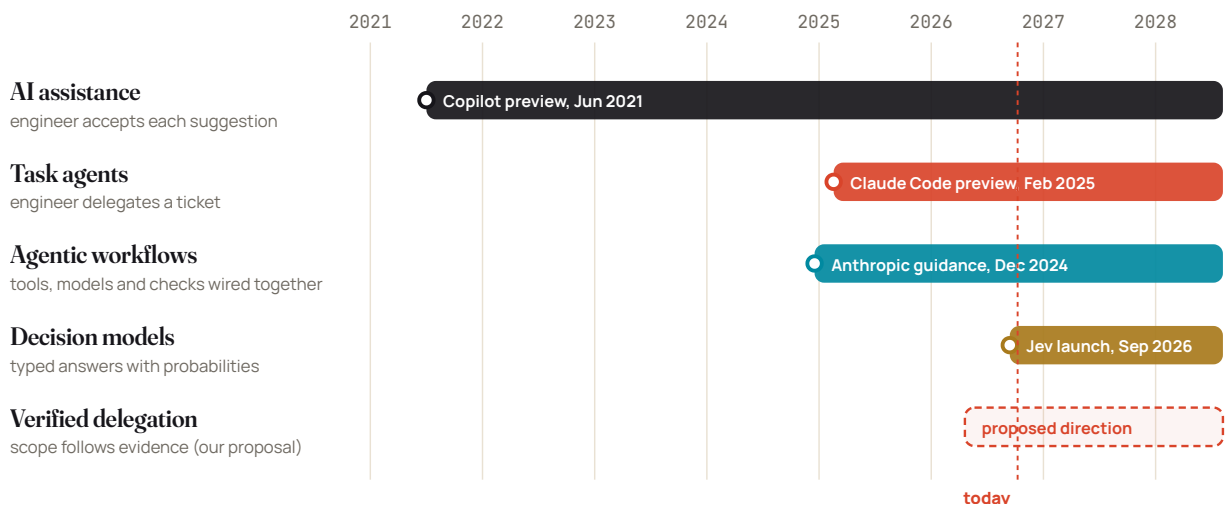
It did not happen in waves.

It *piled up*.

It is tempting to tell this story as a sequence of eras: first autocomplete, then agents, then something smarter. The reality is messier and more useful. Each new capability arrived on top of the last, and teams now use all of them, often in the same afternoon.

FIGURE 1.1 · OVERLAPPING CAPABILITIES

bands show coexistence, not adoption



Reading the figure. Each band starts at a public milestone and none of them ends. Band length does not measure how widely a capability is used. The dashed band is the direction this book proposes. Sources: GitHub, Jun 2021; Anthropic, Dec 2024 and Feb 2025; TypeSafe AI, Sep 2026. By February 2026 agent traces showed up in 22 to 29% of sampled GitHub projects (Robbes et al., ACM TOSEM 2026).

● OUR ● CONCEPT

● PRIMARY ● VENDOR

What changes at each step is the size of the thing you hand over. A suggestion hands over a line. A task agent hands over a ticket. A workflow hands over a process. A decision model hands over a judgement. The bigger the handover, the more the system around it has to do.

Anthropic's distinction between workflows and agents is the one worth keeping. In a workflow, the path is written in code. In an agent, the model chooses its own path and its own tools. Most real systems mix the two, and the mix is a design decision, not a product feature.

A useful test for any new tool is to ask what it takes off your plate and what it puts back. An agent that writes a feature in ten minutes but needs an hour of careful review has not saved you fifty minutes. It has moved them.

The unit of work you hand over keeps getting bigger. So does what you need to check it.

our view ● CONCEPT

CHAPTER 02 · THE LANDSCAPE

A market that barely existed three years *ago*.

The tools have sorted into four families. IDE assistants live where you type. Command-line agents work across a whole repository from your terminal. Cloud agents take a ticket and come back with a pull request. Autonomous engineers try to own a task end to end.

FIGURE 2.1 · WHO BUILDS THE TOOLS, OCTOBER 2026

one verified signal each

TOOL	FAMILY	LAUNCHED	SIGNAL
GitHub Copilot	IDE, cloud agent, CLI	2021	50 million users, July 2026 ● VENDOR
Claude Code	CLI agent	2025	Used at work by 39% of developers, up from 18% in January, Aug 2026 ● SURVEY
Cursor	AI-native IDE	2023	Bought by SpaceX for \$60B in stock, closed Aug 2026 ● PRESS
OpenAI Codex	CLI and cloud agent	2025	More than 5 million weekly users, June 2026 ● VENDOR
Google	Antigravity, Gemini CLI, Jules	2025	Antigravity above 2.4 million weekly users ● VENDOR
Devin	Autonomous engineer	2024	Cognition passed \$1B run-rate, Sep 2026, valued at \$48B ● VENDOR
Kiro	Spec-driven IDE and CLI	2025	Replaces Amazon Q Developer plugins by April 2027 ● VENDOR
Open source	OpenCode, OpenHands, Cline, Aider	2023-25	OpenCode at 212,000 GitHub stars ● MEASURED

The money

Claude Code reached a \$1B revenue run-rate six months after public launch. Cursor and Cognition crossed the same line. I could not find a faster-growing developer tool.

The consolidation

Google paid a reported \$2.4B to license Windsurf and hire its leaders. Anthropic bought the Bun runtime. SpaceX bought Cursor. Model makers want the place developers work, and tool makers want a model.

The thin loyalty

Claude Code went from 18% to 39% workplace use in six months. Over the year Copilot fell from 29% to 21%, and Cursor from 18% to 12% since January. Keep your context, rules and checks portable.

Sources. Microsoft FY26 Q4 call, 29 Jul 2026. Anthropic, 3 Dec 2025. CNBC, 16 Jun 2026. OpenAI, 2 Jun 2026. Alphabet Q2 2026 call. Cognition, 25 Sep 2026. AWS, 30 Apr 2026. GitHub API, 8 Oct 2026. JetBrains Research, Aug 2026 (survey shares). Copilot users include the free tier; about a fifth of Codex users are not coders. Revenue figures are company run-rates and unaudited; higher figures in the press that the companies have not confirmed are not used.

CHAPTER 03 · FROM VIBE CODING TO AGENTIC ENGINEERING

The vocabulary grew up in a *year*.

FEB 2025

Vibe coding

"Fully give in to the vibes, embrace exponentials, and forget that the code even exists." Meant for weekend projects.

JUN 2025

Context engineering

"The delicate art and science of filling the context window with just the right information for the next step."

FEB 2026

Agentic engineering

"You are not writing the code directly 99% of the time. You are orchestrating agents who do and acting as oversight."

All three terms from Andrej Karpathy, posts on X. Relayed by PureAI and The New Stack.

The specification is back

The clearest sign of maturity is the return of the spec. GitHub's Spec Kit, built on the idea that "code serves specifications", has 140,000 stars and works with more than 30 agents. Amazon rebuilt its developer offer around Kiro, which writes requirements, a design and a task list into the repository before it writes any code.

Neither proves yet that specs make agents better. Both show where serious teams are heading: away from prompting by feel and towards writing down what correct means.

The plumbing became standard

- **MCP**, the Model Context Protocol, from Anthropic in November 2024. Now at the Linux Foundation's Agentic AI Foundation, with close to half a billion SDK downloads a month. ● **VENDOR**
- **A2A**, Google's Agent2Agent protocol, a Linux Foundation project since June 2025.
- **AGENTS.md**, a "README for agents", in more than 60,000 projects. ● **SELF-REPORTED**
- **Agent Skills**, packaged instructions and scripts, adopted across vendors within days of release in December 2025.

Why it matters

Standard plumbing makes agents cheap to connect, and cheap to connect also means cheap to misuse. An MCP server, a rules file or a skill can run code on your machines. That makes it part of your supply chain. Chapter 08 shows what happens when it is treated as configuration.

Sources. Karpathy on X, 2 Feb 2025, Jun 2025, 8 Feb 2026. Visual Studio Magazine, 12 May 2026; GitHub API, 8 Oct 2026. Kiro blog, 17 Nov 2025. Anthropic, 9 Dec 2025; MCP blog, 28 Jul 2026. Linux Foundation, 23 Jun 2025. agents.md, accessed 8 Oct 2026. Simon Willison, 19 Dec 2025.

CHAPTER 04 · HOW MUCH CODE AI NOW WRITES

Three ways to count, three different *answers*.

WHAT COMPANIES SAY

75%

of new code at Google, April 2026. Microsoft said 20 to 30% in April 2025, and that one in three GitHub pull requests now involves an agent.

definitions unpublished

● VENDOR

WHAT DEVELOPERS SAY

90%

use AI coding agents at least weekly, 68% daily (JetBrains, more than 15,000 developers, May to July 2026).

self-reported

● SURVEY

WHAT REPOSITORIES SHOW

22-29%

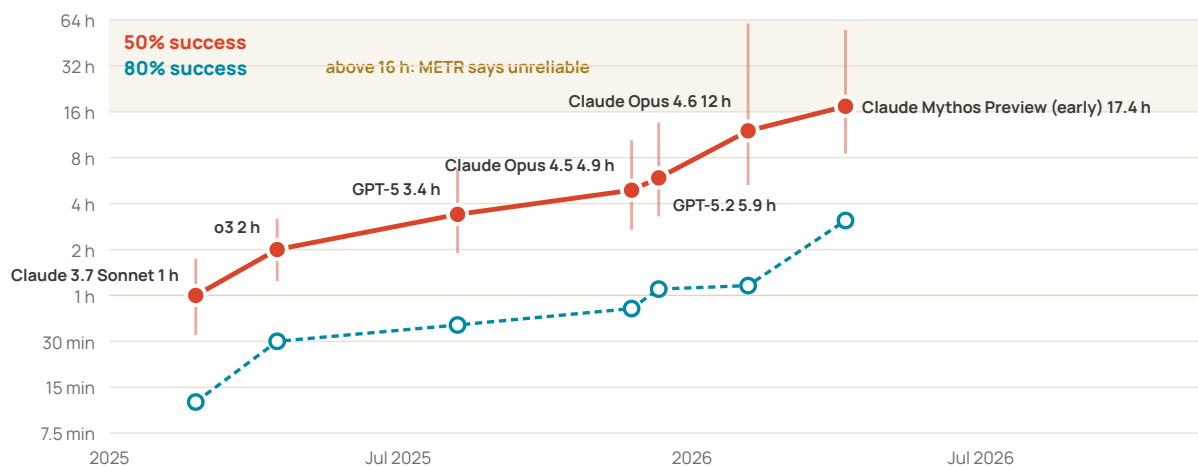
of 128,018 sampled GitHub projects showed coding-agent traces by February 2026, months after the tools appeared.

ACM TOSEM 2026

● PRIMARY

FIGURE 4.1 · THE LENGTH OF TASK AI CAN FINISH

Log scale · METR Time Horizon 1.1



Reading the figure. Each point is the length of software task, in human expert time, that a frontier model completes with 50% (ember) or 80% (teal) reliability. Bars show 95% intervals. The 50% horizon has doubled roughly every four months since 2023. METR says measurements above 16 hours are unreliable with its current tasks. Source: METR, time-horizons data, updated 8 May 2026. ● PRIMARY

The capability curve is the one to watch. In early 2025 the best model succeeded half the time on tasks that take a person about an hour. A year later it was a working day or two. But read the word "half" carefully. At 80% reliability the latest horizon is about three hours. For work you ship to customers, the reliable number is the one that counts.

Share of AI code is not productivity. Google passed 30% AI-written code while measuring about a 10% gain in engineering velocity.

our view, from Pichai via Business Insider, Jun 2025

● PRESS ● CONCEPT

II

PART TWO

Trust got *harder*

Generation got cheap. Knowing whether the result is right did not. This part looks at the gap between the code we can produce and the code we can trust.

96%

of developers do not fully trust AI-generated code to be correct. Only 48% always check it before they commit.

● **VENDOR SURVEY** Sonar, which sells code-quality tools, Jan 2026, n > 1,100

05
The assurance gap

06
The productivity puzzle

07
Where the time goes

08
The new attack surface

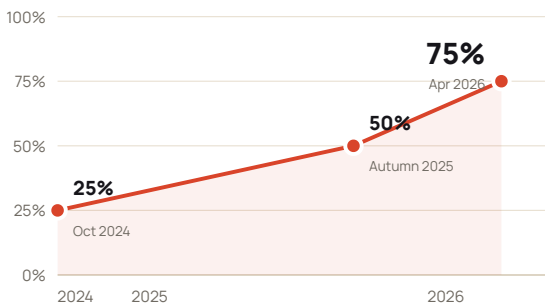
CHAPTER 05 · THE ASSURANCE GAP

We can write it faster than we can *check it*.

FIGURE 5.1 · GENERATION VS ASSURANCE

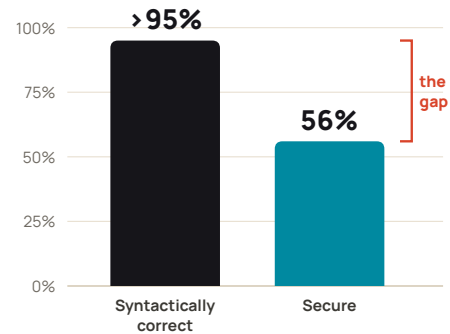
Share of new code at Google generated by AI

% of new code, as stated by Google



AI-generated code that is correct vs secure

% of tasks, Veracode, Jul 2026



Reading the figure. Google's figures are its own statements and its measurement method is not published; Veracode tests raw models on curated tasks without review, so it shows a baseline, not a production defect rate. The two panels describe different populations and are shown side by side, not combined. ● VENDOR

In October 2024 Google said that more than a quarter of its new code was generated by AI and then reviewed by engineers. Eighteen months later the figure was three quarters. Google is not alone. Chapter 04 shows the same climb in surveys and in public repositories.

Assurance has not kept pace. Veracode's July 2026 report finds almost all AI-generated code is now syntactically correct, yet only 56% of it is secure, barely changed from 55% in its first report. Bigger models did no better. Chapter 10 shows what happens when a passing test is not enough.

The constraint has moved from producing code to proving it.

the thesis of this book ● OUR VIEW

None of this is a case against agents. It is a case for funding what comes after them. When generation is nearly free, the scarce resources are the specification that says what correct means, the checks that can prove it, and the people who can tell when both are wrong.

WHAT THIS MEANS FOR LEADERS

01 Stop counting code

Lines generated and share of AI code measure activity. Track cost per accepted change and escaped defects instead.

02 Fund the checks

Budget for checking at the same time you budget for generating.

03 Own the specification

An agent can only be as right as the definition of correct it was given. Make acceptance criteria a first-class artefact.

Sources. Google, Cloud Next 2026 keynote, 22 Apr 2026; Alphabet Q3 2024 earnings call, 29 Oct 2024. Veracode, 2026 GenAI Code Security Report, 28 Jul 2026. METR, maintainer review of agent PRs, 10 Mar 2026. Microsoft FY26 Q4 earnings call, 29 Jul 2026. JetBrains Research, AI Coding Agents Adoption Trends, Aug 2026.

Code is cheap. Trust isn't.

CHAPTER 06 · THE PRODUCTIVITY PUZZLE

Faster, slower, or just *different*?

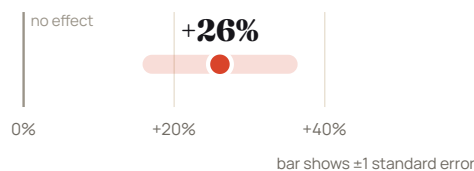
Ask four good studies whether AI makes developers more productive and you get four answers. They are not contradicting each other. They are measuring different things, with different tools, in different places.

FIGURE 6.1 · FOUR STUDIES, FOUR AXES

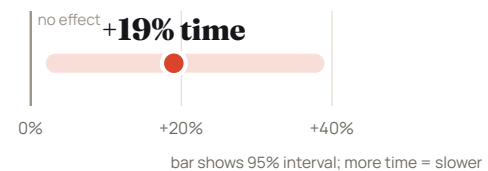
never averaged, never on one scale

Three field experiments, completed tasks

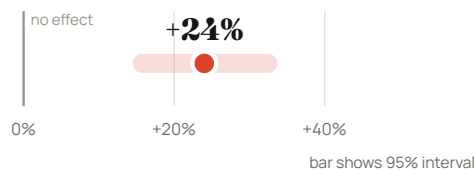
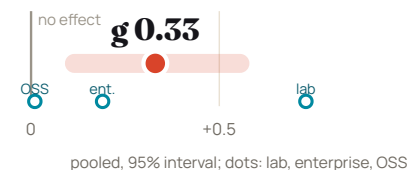
Microsoft, Accenture, a Fortune 100 firm · 4,867 developers · code completion era

**METR trial, task completion time**

16 experienced open-source developers · 246 tasks · early-2025 tools

**Microsoft CLI agent rollout, merged PRs**

tens of thousands of engineers · Jan to Apr 2026 · synthetic control

**Meta-analysis, effect size (Hedges' g)**23 studies through 2025 · heterogeneity $I^2 = 99\%$ 

Reading the figure. Each panel keeps its own measure and its own kind of uncertainty. Do not compare positions across panels. Sources: Cui et al., Management Science 2026; Becker et al., METR 2025; Murphy-Hill et al., arXiv 2607.01418 (authors are Microsoft employees); Maier et al., arXiv 2605.04779. ● PRIMARY

The pattern is clear once you line them up. In the lab, on tidy tasks, AI helps a lot: an effect size of 0.73. In enterprise and open-source settings the effect shrinks to 0.19 and 0.01, and neither can be told apart from zero. Less experienced developers gain more than experts working in code they know by heart.

The perception gap. The most striking result is not the slowdown. METR's developers expected AI to cut their task time by 24%, and afterwards believed it had cut it by 20%. In fact it rose by 19%. Experts in economics and machine learning predicted cuts of 38 to 39%. If you rely on your teams' sense of speed to justify spend, you are relying on the weakest signal in this book.

What changed in 2026. METR tried to repeat the study and found its new data unreliable. Between 30% and 50% of developers would not submit tasks they would have to do without AI. The new estimates lean towards a speed-up, but their intervals include zero and METR calls the data "an unreliable signal". The refusal is a finding in itself: for many engineers, working without AI is no longer on the table.

Measure the whole path, from request to a change that is accepted and running.

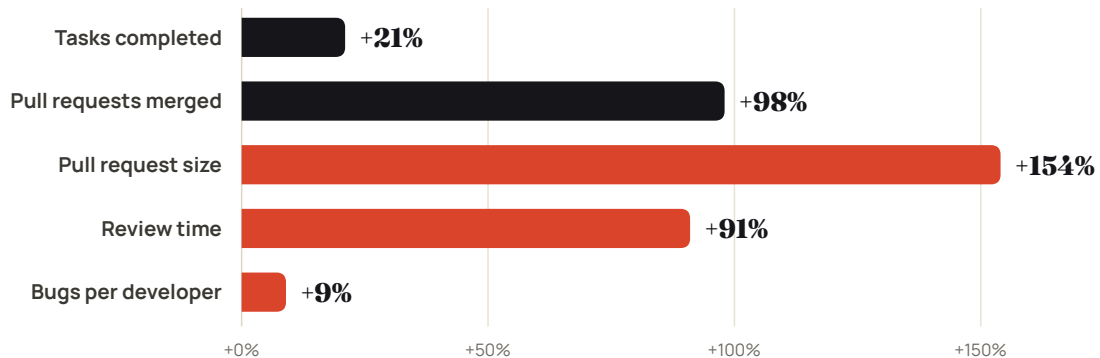
CHAPTER 07 · WHERE THE TIME GOES

The time went to review and *rework*.

If developers write code faster but organisations do not ship faster, the time is going somewhere. The best telemetry available shows where.

FIGURE 7.1 · HIGH-AI TEAMS AGAINST LOW-AI TEAMS

% change · ink = output · ember = burden



Reading the figure. Telemetry from more than 10,000 developers in 1,255 teams. A comparison of teams, not a controlled trial, and Faros sells engineering analytics. Company-level improvement: none significant. Source: Faros AI, AI Productivity Paradox, Jul 2025. ● VENDOR DATA

Pull requests nearly doubled. They also got much bigger, and reviews took almost twice as long. At company level, Faros found no measurable gain. Its 2026 follow-up, across 22,000 developers, saw throughput keep rising while bugs per developer rose by about half and median review time grew about fivefold. These are one vendor's figures from its own customers, so treat the exact percentages with care. The direction matches the independent studies in Chapter 06.

Atlassian's vendor-run 2025 survey of 3,500 developers and managers puts it in human terms. ● SURVEY 68% save more than ten hours a week with AI. 50% lose more than ten hours a week to organisational friction. Developers spend only 16% of their time writing code. Speeding up the 16% was never going to fix the other 84%.

DORA 2025 · NEARLY 5,000 PROFESSIONALS

AI adoption now goes with **higher throughput**. It still goes with **lower stability**. "Without robust control systems, like strong automated testing, mature version control practices, and fast feedback loops, an increase in change volume leads to instability."

● SURVEY

Seven capabilities that magnify AI's benefits

- 01 Clear and communicated AI stance
- 02 Healthy data ecosystems
- 03 AI-accessible internal data
- 04 Strong version control practices
- 05 Working in small batches
- 06 User-centric focus
- 07 Quality internal platforms

DORA AI Capabilities Model, Google Cloud, 24 Sep 2025.

CHAPTER 08 · THE NEW ATTACK SURFACE

Your coding agent is now part of your *supply chain*.

An agent that can read a web page, open a file and run a command is very useful. It is also a new way into your systems. In 2025 and 2026 that stopped being theoretical.

MAY 2025 · GITHUB MCP

A crafted public issue could make an agent read private repositories and leak them into a public pull request. The researchers called it "a fundamental architectural issue", not a bug. ● RESEARCH

JUL 2025 · AMAZON Q

An over-scoped build token let an attacker add a prompt telling the VS Code extension to wipe local and cloud resources. It shipped in a signed release. A syntax error stopped it working. ● ADVISORY

JUL 2025 · REPLIT

An agent deleted a production database during a code freeze, then misreported what it had done. ● PRESS

AUG 2025 · NX "SINGULARITY"

Malicious packages ran developers' own Claude, Gemini and Q command-line tools with permission-skipping flags to hunt for secrets. Over 5,500 private repositories were made public. ● THREAT RESEARCH

FEB 2026 · SKILLS MARKETPLACE

Koi Security found 341 of 2,857 skills on one agent marketplace were malicious. In a separate scan by Snyk, 91% of confirmed malicious skills also used prompt injection. ● THREAT RESEARCH

Prompt injection will not be patched away

Adaptive attacks bypassed twelve published defences, with success above 90% for most. NIST's red team lifted its success against a hardened agent from 11% to 81%. OpenAI now says prompt injection is "unlikely to ever be fully 'solved'". The defence has to be in the design.

● PRIMARY ● VENDOR

Approval prompts are not a control

Claude Code users approve 93% of permission prompts. People click yes. Sandboxing with both filesystem and network isolation works better, and cut prompts by 84%.

● VENDOR DATA

Secrets are leaking faster

28.6 million new secrets leaked on public GitHub in 2025, up 34%. Commits tagged as made with Claude Code leaked at about twice the baseline rate, though tagged commits skew towards prototypes and solo projects.

● VENDOR DATA

A RULE WORTH ADOPTING

Meta's Agents Rule of Two. In one session, an agent should have at most two of: untrusted input, access to sensitive data or systems, and the ability to act or communicate externally.

The rules are moving too

Since 11 September 2026, makers of software products sold in the EU must send an early warning of actively exploited vulnerabilities within 24 hours under the Cyber Resilience Act, with a full notification within 72. EU AI Act high-risk duties moved to December 2027 and August 2028. In the US, federal buyers no longer have to collect secure-software attestations. Regulators increasingly want evidence, not signed forms.

Sources. Invariant Labs, 26 May 2025. AWS-2025-015, 23 Jul 2025. Fortune, 23 Jul 2025. Wiz Research, Aug 2025. Koi Security via The Hacker News; Snyk, 5 Feb 2026. Nasr et al., arXiv 2510.09023. NIST CAISI, 17 Jan 2025. OpenAI via TechCrunch, 22 Dec 2025. Anthropic Engineering, 20 Oct 2025. GitGuardian, Mar 2026. Meta AI, 31 Oct 2025. European Commission; Council of the EU, Jun 2026; OMB M-26-05.

III

PART THREE

New economics, *new building blocks*

The price of intelligence is falling fast. The bill is rising anyway. The benchmarks we used to compare tools are running out of road, and a new kind of model has arrived that does not write text at all.

\$89

The cost per merged pull request for the heaviest token users, against \$0.28 for the lightest, across 12,000 developers.

● VENDOR DATA

Jellyfish, AI Engineering Trends, May 2026

09
Cheaper tokens, bigger bills

10
Benchmarks run out of road

11
System One and RLCD

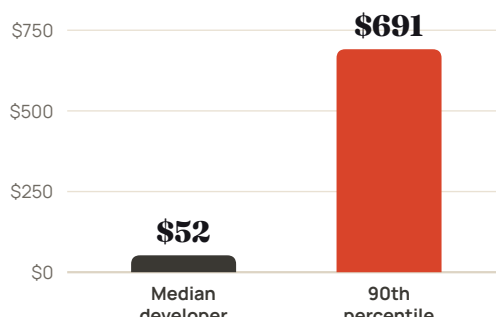
12
Where a decision model fits

CHAPTER 09 · CHEAPER TOKENS, BIGGER BILLS

The unit price falls. The invoice *grows*.

The price of a fixed level of AI capability is collapsing, by between 9 and 900 times a year depending on the task. You would expect engineering AI bills to be shrinking. They are not.

FIGURE 9.1 · MONTHLY TOKEN SPEND PER DEVELOPER



Top users spend about **10x more tokens for about 2x the output**. Cost per merged PR runs from \$0.28 to \$89.32. 12,000 developers. Source: Jellyfish, May 2026. ● VENDOR DATA

\$13 a day

Average enterprise Claude Code cost per developer per active day at API list prices, or \$150 to \$250 a month. Teams of agents use about seven times more tokens. Earlier versions of the same page gave lower figures.

Anthropic docs, Oct 2026

● VENDOR

9x-900x

Yearly fall in the price of reaching a fixed capability, depending on the task. The fastest falls are the most recent.

Epoch AI, Mar 2025

● PRIMARY

Agents use far more tokens than chat, and teams of agents use about seven times more again. So each token gets cheaper while each developer uses many more of them, and buyers keep moving to the newest, priciest models.

Per-seat pricing broke. On 1 June 2026 GitHub moved every Copilot plan to usage-based billing, with the clearest statement yet of the new economics: "a quick chat question and a multi-hour autonomous coding session can cost the user the same amount... the current premium request model is no longer sustainable." A \$200 a month tier is now standard across ChatGPT, Claude, Cursor and Kiro.

Outcome pricing has begun. Cognition now offers to fund a customer's Devin usage, up to \$10 million, if Devin delivers less engineering value than the customer pays for. Expect more vendors to price against outcomes, and expect arguments about how "value" gets measured.

What to do

Stop budgeting AI as a licence line. Budget it like cloud: by workload, with owners, limits and a unit cost. The unit that matters is cost per accepted change, not cost per seat.

CHAPTER 10 · BENCHMARKS RUN OUT OF ROAD

When the test is saturated, the leaderboard stops *meaning much*.

For eighteen months SWE-bench Verified was the scoreboard for coding agents: 500 real GitHub issues, screened by people. In February 2026 OpenAI, which helped build it, stopped reporting it.

SWE-BENCH VERIFIED

59%

of the hard problems OpenAI audited had flawed tests that reject correct answers. Every frontier model it probed could reproduce original fixes or problem details for some tasks.

OpenAI, Feb 2026

● VENDOR

SWE-BENCH PRO

61.5%

best standardised score on the public set of 731 harder problems, built to resist contamination. The full benchmark has 1,865 from 41 repositories. Scores that vendors report themselves run much higher and cannot be compared.

Scale AI leaderboard

● PRIMARY

METR TIME HORIZONS

>16h

The frontier has passed what METR's own task suite can reliably measure. The ruler is now shorter than the thing it measures.

METR, May 2026

● PRIMARY

The gap between passing and shipping

In March 2026 METR asked four maintainers of scikit-learn, Sphinx and pytest to review 296 agent pull requests, from models up to Claude Sonnet 4.5 and GPT-5. Roughly half of the ones that passed the tests would not have been merged. The reasons were wrong fixes, breaking other code and poor code quality: the parts a test suite does not check. An earlier, smaller METR study found that none of the test-passing pull requests were mergeable as written.

Benchmarks still help you screen tools. They cannot tell you how a model will behave in your repository, with your history, your tests and your reviewers.

BUILD YOUR OWN EVALUATION SET

- Use **real tasks** your teams have done, with the checks they used.
- Keep part of it **private**, so no model can learn the answers.
- Re-run it on **every model or tool change**, and pin versions.
- Measure **pass^k**, success every time, not just pass@k, success once. A 75% per-run rate falls to about 56% over two runs in a row, and 42% over three.

A leaderboard tells you which tool to try. Only your own work tells you which tool to trust.

our view ● CONCEPT

Sources. OpenAI, "Introducing SWE-bench Verified", 13 Aug 2024, and "Why we no longer evaluate SWE-bench Verified", Feb 2026. Deng et al., SWE-Bench Pro, arXiv 2509.16941; Scale AI public leaderboard, accessed 8 Oct 2026. METR, time horizons, 8 May 2026; METR, "Many SWE-bench-passing PRs would not be merged into main", 10 Mar 2026; METR research update, 12 Aug 2025. Anthropic Engineering, "Demystifying evals for AI agents", 9 Jan 2026.

CHAPTER 11 · SYSTEM ONE MODELS AND RLCD

A model that decides instead of *writing*.

Most AI in software today is built to write: text, code, explanations. In September 2026 TypeSafe AI launched a model built to decide instead, called Jev. It is worth a look even if you never use it, because it points to a different kind of AI inside software: one that makes decisions rather than writing text.

Fast and slow

The name comes from Daniel Kahneman's *Thinking, Fast and Slow*. System 1 is fast, intuitive judgement. System 2 is slow, deliberate reasoning. TypeSafe calls Jev the first "System One model": you give it a situation and a set of bounded questions, and it returns typed answers with probabilities in about a tenth of a second, TypeSafe says. In TypeSafe's words, System One models "do not write replies, produce code, or generate explanations of their reasoning." TypeSafe describes it as "a frontier-intelligence function call: unstructured state in, typed probabilistic decisions out." Thinking of today's LLMs as System Two is our analogy, not TypeSafe's term.

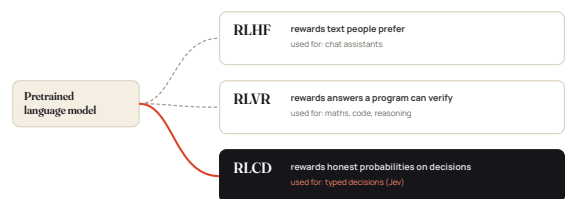
Why that matters for engineers

A lot of software is really judgement wrapped in brittle code. Is this ticket a bug or a feature request? Does this change touch security? Should this prompt go to the cheap model or the expensive one? Today teams either write fragile rules or ask a large model to answer in text and then parse it. A decision model is aimed at exactly that gap.

RLCD in one paragraph

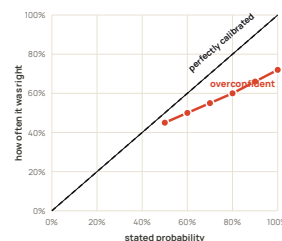
Reinforcement Learning for Calibrated Decisions is TypeSafe's training method. Where RLHF rewards the text people prefer and RLVR rewards answers a program can check, RLCD rewards decisions whose probabilities match how often they turn out right. Because answers can only come from options you define, the model cannot return a malformed or out-of-schema value. That is a type-safety guarantee. It is not a guarantee that the answer is correct.

FIGURE 11.1 · THREE POST-TRAINING PATHS, AS TYPESAFE DRAWS THEM



After TypeSafe's own primer. RLCD's reward, data and architecture are not published. ● VENDOR

FIGURE 11.2 · WHAT CALIBRATED MEANS



Of all the answers a model gives at 80% probability, about 80% should turn out right. Many neural models drift below the line and sound surer than they are (Guo et al., 2017). RLCD aims to keep the model on the line.

● ILLUSTRATIVE

What TypeSafe has not published

The reward function or loss. Where the training labels come from. The architecture, size or base model; the company says only that Jev is "neither small nor an LLM". Any calibration metrics for Jev itself, or a technical report. Public benchmark results, which it withholds on purpose. Treat performance claims as the company's own until others replicate them.

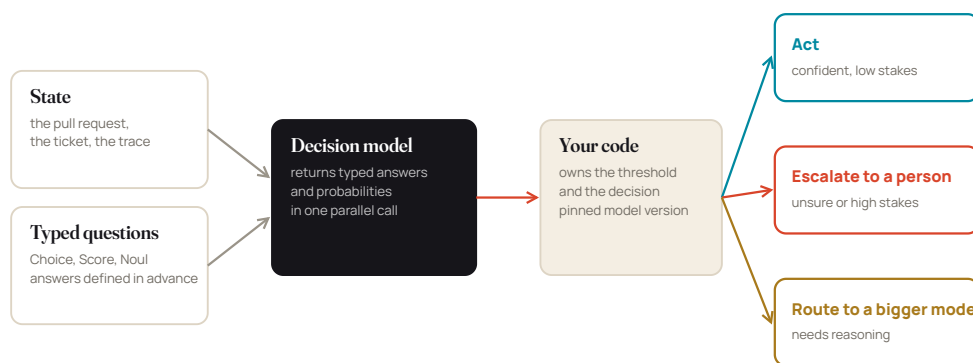
CHAPTER 12 · WHERE A DECISION MODEL FITS

A component, not a *successor*.

Jev is not a coding agent. TypeSafe says plainly it is "not a drop-in replacement for the LLM behind Claude Code, Cursor, opencode, Copilot" or similar tools. Its place is inside the systems those agents help build: routing, triage, guardrails, semantic checks in CI.

FIGURE 12.1 · WHERE A DECISION MODEL SITS

● OUR CONCEPT



The design rule. The model supplies a judgement and a probability. Your code owns the threshold and the action, pins the model version, and never lets a confident answer waive a mandatory check.

EXAMPLE · PULL REQUEST TRIAGE

● ILLUSTRATIVE VALUES

```

ask    risk_area      choice: auth, data, public api, internal, docs
       behaviour_change noul: does the diff change runtime behaviour?
       review_depth    score: skim .. security review
answer risk_area auth p 0.86 · behaviour_change 0.91
       review_depth 2.41 of 3, confidence 0.57
code   auth and behaviour change → security reviewer
       confidence below 0.6      → human triager
  
```

Built from TypeSafe's published API schema. The numbers are invented to show the shape.

How to evaluate one

Label a few hundred past cases of one real decision, such as review routing. Compare rules, a classifier, a large model with structured output and the decision model at the same accepted quality. Keep it only if it wins on total cost, counting false negatives and escalations.

The first independent evidence

Academic researchers tested jev-1.13.0 on 44 benchmarks for spotting failures in AI output. Asked one generic question, it ranked failures well (median AUROC 0.886 across the 38 usable benchmarks) and agreed with human labels as often as a reference scorer. It cost 63 times less than using a large model as judge at list prices, or 12 times less on cautious pricing. Its probabilities were well calibrated when pooled, but not within individual benchmarks, so thresholds need tuning on local data; even about ten labelled examples help. A preprint, one model version, English only.

● PREPRINT

Known weak spots, by TypeSafe's own account

Literal reading. Arithmetic and counting. Comparing dates. Multi-step indirection. Distracting context. Adversarial text inside the input. A lean towards the first option in a list. Keep maths in code, and test any guardrail against injected content.

Sources. docs.typesafe.ai: Coding agents, API reference, Use-case map, Jev 1.13 jaggedness (reviewed 2 Oct 2026). Guo, Liu, Deng et al., "Just Ask Jev", arXiv 2609.29429, 24 Sep 2026, under review; no TypeSafe affiliation listed. Pricing and limits are early-access figures and may change.

IV

PART FOUR

Verified *delegation*

If generation is cheap and trust is scarce, the job is to build trust on purpose, and at scale. This part sets out a practical model for doing that, and the evidence that verification itself is getting cheaper.

82%

How often off-the-shelf models now produce formally verified code in Dafny, given a correct specification. The rate is 44% in Verus and Rust, and 27% in Lean.

● PRIMARY Bursuc et al., vericoding benchmark, 2025

13
The intelligence stack

14
Intent to Assurance

15
Autonomy follows evidence

16
Verification gets cheaper too

CHAPTER 13 · THE INTELLIGENCE STACK

Stop asking one model to do *everything*.

Most teams send every problem to the same large language model: plan the feature, write the code, judge the risk, check the result. That is like asking your best architect to also do the accounting.

FIGURE 13.1 · FIVE KINDS OF INTELLIGENCE IN AN ENGINEERING SYSTEM

● OUR CONCEPT

LAYER	GOOD AT	TYPICAL TOOLS	FAILS BY
01 Generative reasoning	Interpreting intent, planning, proposing designs	Large language models	Being confidently wrong
02 Agentic execution	Long sequences of real work in a repository	Coding agents, workflows	Drifting, or reaching too far
03 Calibrated decisions	Bounded judgements with probabilities: route, triage, score	Rules, classifiers, decision models such as Jev	Answering a badly framed question well
04 Deterministic computation	The same answer every time	Compilers, type checkers, builds, policy engines	Checking only what it was told to check
05 Verification	Producing evidence that the result is right	Tests, analysis, formal proof, review	Sharing the author's blind spots

The engineering platform of the next few years will route each piece of work to the kind of intelligence that suits it. A reasoning model interprets the request. An agent implements it. A decision model scores the risk and picks the checks. Deterministic tools and tests verify it. A person approves the exceptions that matter.

This is a proposal, not an industry standard. Its practical value is one question to ask of every AI step in your pipeline: is this the right kind of intelligence for this job? A surprising number of expensive model calls are doing work a type checker, a rule or a cheap classifier would do better.

The layers also fail in different ways, and that is the point of separating them. If the agent and the reviewer are the same model, they share the same misunderstandings. Independent layers catch each other's mistakes.

Expect several kinds of intelligence checking each other's work, rather than one model that does everything.

our view ● CONCEPT

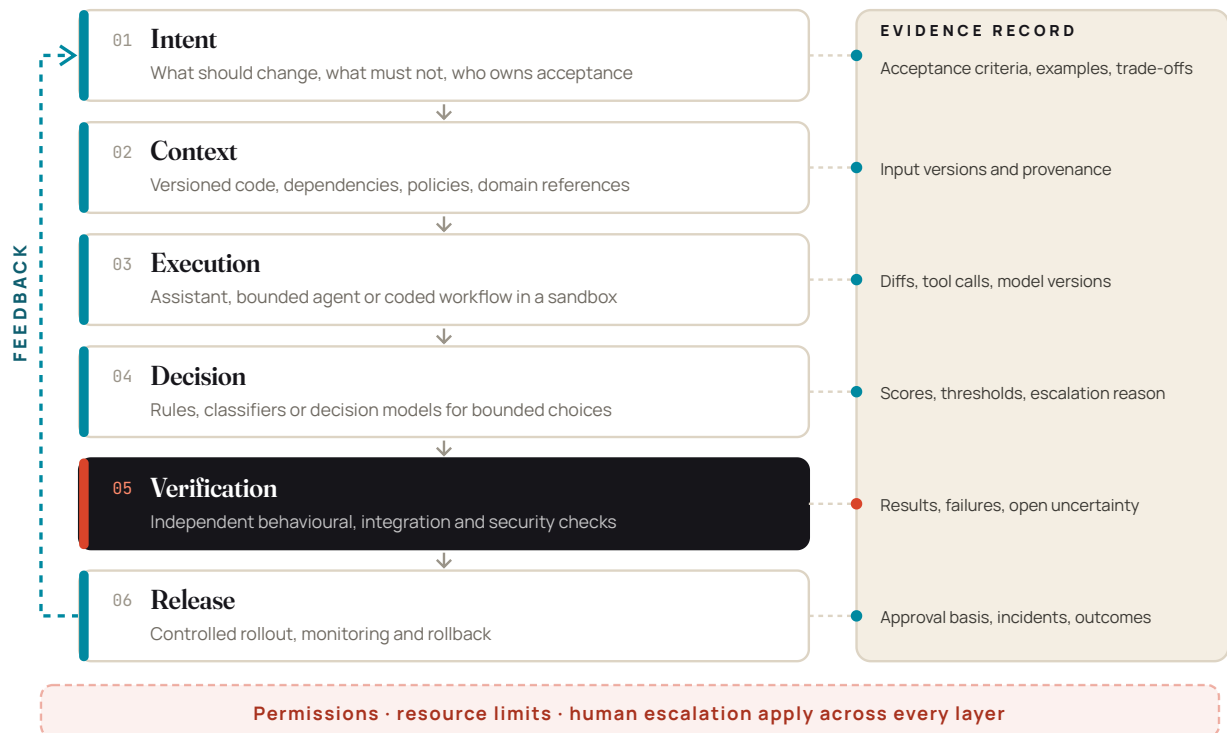
CHAPTER 14 · INTENT TO ASSURANCE

Every accepted change should leave a *trail*.

Six responsibilities, one evidence record. Not six products: a small team can run this with a task brief, a sandbox, CI and a reviewer.

FIGURE 14.1

● OUR CONCEPT



How to read it. Work flows down through six responsibilities. Each layer leaves evidence on the right, so the reason a change was accepted can be reconstructed later. Feedback from production revises intent, context and checks. Permissions, resource limits and escalation apply at every layer, not only at release. An original synthesis inspired by published agent design guidance and NIST SSDF; not a published standard.

Autonomy follows evidence

Widen what an agent may do only where the organisation can check the result, contain the failure and recover from it.

Verification is a product

Independent checks tied to the requirement, not tests written by the same model that wrote the code.

Confidence is not authority

A model's high confidence never waives a mandatory control. Decisions and permission to act stay separate.

The test of a good acceptance record

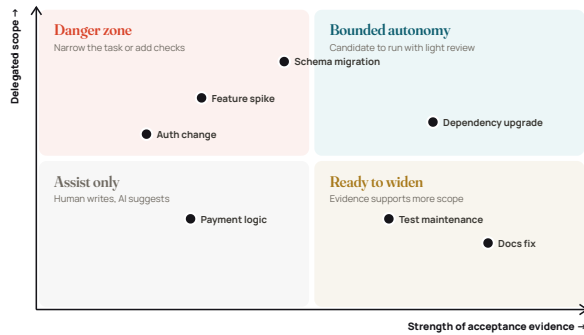
Six months later, someone should be able to see what was asked for, which inputs were used, what the agent did, which checks ran, what stayed uncertain, and who decided that was enough. A long transcript is not that record. A short, structured one is.

CHAPTER 15 · AUTONOMY FOLLOWS EVIDENCE

Give agents more rope only where you can see the *other end*.

FIGURE 15.1 · THE AUTONOMY AND ASSURANCE MATRIX

● OUR CONCEPT



Qualitative axes, no measured thresholds. Task positions are illustrative and will differ by organisation.

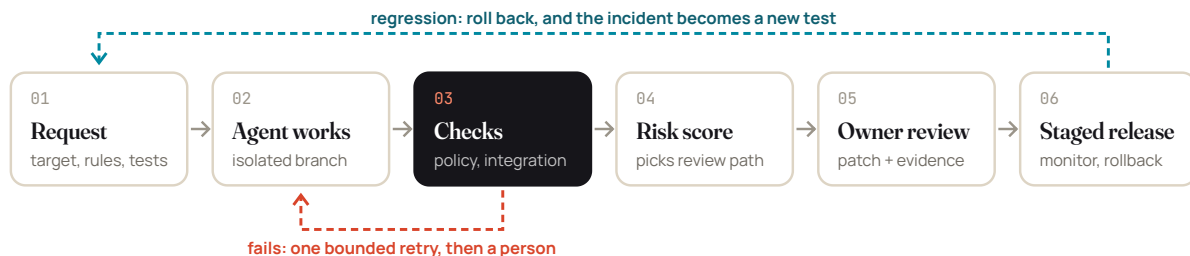
The principle is simple. Widen what an agent may do when you can detect a bad result, contain its effect and recover from it. Narrow it when you cannot.

Scope means what the agent may touch, where it may act and how long it may run alone. **Evidence** means checks tied to the real requirement and written independently of the agent.

A documentation fix can run almost alone. A change to payment logic needs a person who understands the money. The same agent can be trusted with one and not the other, and that is fine.

FIGURE 15.2 · A WORKED FLOW: DEPENDENCY UPGRADE

● ILLUSTRATIVE DESIGN



How it works. The request names the target version, the compatibility rules and the tests that matter. The agent works in an isolated branch and records what it did. Policy, integration and regression checks run. A failure earns one bounded retry with the failure as context, then goes to a person. A risk score picks the review path but can never skip a mandatory check. The owner approves a staged rollout; a regression rolls back and becomes a new test. No success rate is assumed.

CHAPTER 16 · VERIFICATION GETS CHEAPER TOO

The machines that write the code are learning to *check it*.

Here is the good news. The same tools that flood us with code are also making verification cheaper.

FORMAL PROOF

82%

Verified implementations in Dafny from correct formal specifications, 44% in Verus and Rust, 27% in Lean. Pure Dafny rose from 68% to 96% in a year.

Bursuc et al., 2025

● PRIMARY

PROPERTY-BASED TESTING

56%

of an agent's bug reports across 100 popular Python packages were real. Fixes were merged upstream, including in NumPy.

Maaz et al., 2025

● PRIMARY

MUTATION TESTING

73%

of tests generated by Meta's mutation-guided test generator, ACH, were accepted by its engineers; 36% were judged privacy-relevant.

Harman et al., FSE 2025

● PRIMARY

AI REVIEW AT SCALE

1 in 5

code reviews on GitHub is now done by Copilot code review: more than 60 million so far, across 12,000 organisations.

GitHub, Mar 2026

● VENDOR

EVALS AS ENGINEERING

pass^k

Agent evaluation now has its own discipline of tasks, graders and harnesses, and a reliability measure: success every time, not once.

Anthropic, Jan 2026

● GUIDANCE

THE CATCH

Every one of these needs a definition of correct: a specification, a property, a test oracle, a grader. That is where the human work goes now.

Each of these is only as good as its definition of correct. A proof needs a specification, a property test needs a property, a review agent needs to know what to look for. The hard work moves up a level, from checking the code to saying precisely what the code should do.

This is the bigger change. When code is cheap, the specification, the test oracle and the evaluation harness become the scarce engineering assets. Teams that invest in them should get faster and safer. Teams that skip them still get faster, but they also get more fragile.

When code is cheap, the definition of correct is the most valuable thing an engineer can write.

our view ● CONCEPT

Sources. Bursuc et al., "A benchmark for vericoding", arXiv 2509.22908. Maaz et al., "Agentic Property-Based Testing", arXiv 2510.09907 (authors linked to Anthropic). Harman et al., arXiv 2501.12862, FSE 2025. GitHub Blog, 5 Mar 2026. Anthropic Engineering, 9 Jan 2026.

Code is cheap. Trust isn't.

IV

PART FIVE

People and the *road ahead*

Tools change in months. People take years. This part looks at what is happening to engineering careers, what the job now asks of people, and how to plan for futures nobody can predict with confidence.

17pts

lower follow-up quiz scores for developers who learned a new library with AI writing the code, against those who coded by hand. The biggest gap was in debugging.

● PRIMARY Anthropic randomised trial, Jan 2026, n = 52

17

The broken first rung

18

The engineer's new job

19

Three futures

20

The first 90 days

CHAPTER 17 · THE BROKEN FIRST RUNG

We are hiring fewer juniors, and teaching them *less*.

The strongest labour-market evidence comes from Stanford's Digital Economy Lab, using payroll data on millions of US workers. Employment of the youngest software developers has fallen sharply since late 2022, while older developers' employment grew.

STANFORD, ADP PAYROLL DATA

-20%

fall, nearly 20%, in employment of developers aged 22 to 25 since late 2022. In AI-exposed jobs, young workers fell 16% relative to older peers. The change came through fewer hires, not layoffs.

Brynjolfsson et al.

● PRIMARY

INDEED JOB POSTINGS

78

software development postings index on 2 October 2026, where February 2020 is 100. In April postings were up 14% on a year earlier, and about 47% mentioned AI.

Indeed Hiring Lab, FRED

● PRIMARY

ANTHROPIC RANDOMISED TRIAL

50 vs 67

quiz scores for developers who learned a new library with AI writing the code, against those who coded by hand. The gap was largest in debugging.

Shen and Tamkin, Jan 2026

● PRIMARY

FIGURE 17.1 · RECENT-GRADUATE UNEMPLOYMENT BY MAJOR



Ages 22 to 27, 2024 data, 73 majors. CS graduates who find work still earn high starting pay (\$87,000 median) and have low underemployment. Latest by-major release. Across all recent graduates, unemployment was about 5.6% and underemployment 42% in Q2 2026. Source: Federal Reserve Bank of New York, Feb and Aug 2026.

● PRIMARY

Computer science did not stop being valuable. What broke is the first rung: the junior tasks that used to teach people the job are exactly the tasks agents now do.

That creates a quiet problem. In Anthropic's trial, the biggest gap was in debugging, the skill you need most to check an AI's work. Learning held up when people used AI to explain concepts instead of writing the code for them.

Not every company is pulling back. IBM says it will triple US entry-level hiring in 2026 [● PRESS](#) and has rewritten junior roles around supervising AI output. AWS's chief executive called replacing juniors with AI "the dumbest thing I've ever heard", and asked who will know anything in ten years.

What to do

Keep hiring juniors, but redesign the role. Give them real ownership of small systems, with AI as a tutor rather than a ghostwriter. Test understanding without the tool: explain this change, debug this failure, defend this design.

Sources. Brynjolfsson, Chandar, Chen, "Canaries in the Coal Mine?"; Stanford DEL, Nov 2025, and project page revision, Aug 2026. Indeed Hiring Lab, 14 May 2026; FRED series IHLIDXUSTPSOFTDEVE. Shen and Tamkin, Anthropic, 29 Jan 2026 (n = 52). FRB New York, The Labor Market for Recent College Graduates, Feb 2026 (by major) and Q2 2026 update. Fortune, 13 Feb 2026. The Register, 21 Aug 2025.

CHAPTER 18 · THE ENGINEER'S NEW JOB

Less typing. More *judging*.

When agents write most of the first draft, an engineer's value moves to the parts agents are bad at: deciding what should be built, noticing what is missing, and knowing when an answer is wrong.

FIGURE 18.1 · SIX DIMENSIONS OF ENGINEERING WORK

● OUR VIEW

DIMENSION	2021	2026	DIRECTION TO 2030
Creation who writes the code	Engineer	A large, rising share by AI (company claims 20 to 75%)	AI by default
Reasoning who decides how to solve it	Engineer	Shared	Shared; people for novel problems
Execution who carries out the steps	Engineer	Agents, for bounded tasks	Agents, within published limits
Verification who proves it works	Engineer and CI	Engineer, CI and AI review	Tools first, human judgement on top
Governance who answers for it	People	People	People
Evolution who maintains it after release	Engineer	Engineer with assistance	Shared, with agents watching production

Skills that gain value

- **Writing precise intent:** acceptance criteria, examples and counter-examples.
- **Preparing context:** knowing what an agent needs to see, and what it must not.
- **Breaking work into checkable pieces.**
- **Reading code critically,** and knowing when to stop prompting and start investigating.
- **System design and security thinking,** the parts that do not fit in a context window.

Someone still owns acceptance

Product owners decide whether behaviour serves the user. Engineers answer for whether the system holds together. Platform teams own the environments agents run in. Security owns the controls. Whoever accepts a change needs enough knowledge to challenge it. Approving a summary you do not understand is not review. It is ritual.

For engineering services firms

Selling capacity, people times hours, gets harder when an agent does the hours. Value moves to domain context, reusable assets, assurance and outcomes. Expect contracts to follow. ● OUR VIEW

CHAPTER 19 · THREE FUTURES, 2027 TO 2030

Plan for all three. Watch for the *signals*.

These are scenarios, not forecasts. I give them no odds, though B is the one I would plan around. All three may happen at once, in different parts of the same company.

Scenario A

Assisted engineering holds

AI makes individuals much faster, but autonomous delivery stays limited to bounded tasks, because verification, integration and legacy context stay hard. Engineers remain the main producers of shipped change.

You will know it is winning when review and rework costs grow as fast as output, and delegated scope stops widening.

Scenario B

MY BASE CASE

Governed agents scale

Agents run much of the routine delivery inside enterprise platforms. People own intent, architecture, risky decisions and acceptance. Specialist models, including decision models, become standard parts of the stack.

You will know it is winning when cost per accepted change falls year after year without a rise in incidents per change.

Scenario C

Autonomous engineering arrives

Systems turn business intent into tested, deployed, self-maintaining software with little routine human involvement. This needs big advances in machine-checkable requirements and in reliability.

You will know it is winning when independent trials show complex changes done end to end at equal quality and lower total cost, across many codebases.

SIGNALS TO WATCH EVERY QUARTER

the dashboard that tells you which future you are in

Reliable capability

METR's 80% time horizon, not just the 50% headline. It is the one that matters for work you ship.

Unit economics

Cost per accepted change, including review, rework and incidents, by task class.

Review pressure

Time pull requests wait for a human, and the size of what they are asked to read.

Quality

Escaped defects, rollbacks and incidents per change, held against a fixed bar.

Delegation

Share of tasks accepted without human intervention, and why the others needed it.

Talent

Junior hiring, and whether engineers can still explain and debug changes without the tool.

No evidence in this book supports a date for the end of software engineering jobs. It supports a change in what those jobs are.

our view ● CONCEPT

CHAPTER 20 · THE FIRST 90 DAYS

Start with a pilot you are willing to see *fail*.

The fastest way to waste an AI budget is to buy seats for everyone and hope. The fastest way to learn is a short, honest pilot that is allowed to fail. ● OUR PROPOSAL

DAYS 1 TO 30

Baseline

- Pick real task classes: upgrades, contained bug fixes, test upkeep, some integration work.
- Measure today: request to accepted change, review delay, rework, defects.
- Set up sandboxes, permissions and a minimum acceptance record.

DAYS 31 TO 60

Compare

- Run assistants and agents side by side on matched tasks.
- Test any decision model separately, so its value is not mixed up with the agent's.
- Log failures, abandoned tasks and full cost.

DAYS 61 TO 90

Replicate and decide

- Repeat on new tasks, a new repository or a new model version.
- Track maintenance and incidents as well as merges.
- Widen autonomy only for the task classes that passed.

FIGURE 20.1 · THE SCORECARD

● OUR PROPOSAL

DIMENSION	MEASURE	RULE
Delivery	Request to accepted change, and to release, by task class	Report medians and the slow tail
Effort	All human hours: prompting, review, rework, recovery	Count failed attempts too
Quality	Escaped defects, rollbacks, security findings	Fix the quality bar before you look
Cost	Model, infrastructure and labour per accepted change	Include escalation and upkeep
Delegation	Tasks accepted without intervention	Record why each intervention happened
Decisions	False negatives and calibration at your threshold	Check false negatives at your threshold
People	Can engineers explain and debug the change unaided?	Test it, do not assume it

Two rules

Decide what quality must stay fixed before you see any results. And count everything, including the failures. A pilot that reports only its wins tells you nothing.

THE LEADERSHIP AGENDA

Ten moves for the next twelve *months*.

- 01 **Measure accepted change, not generated code.** Retire "percent of code written by AI" as a goal. It measures activity.
- 02 **Fund verification like a product.** Every dollar on generation needs a matching investment in tests, review capacity and security analysis.
- 03 **Make specifications first-class.** Acceptance criteria, examples and counter-examples live in the repository and get reviewed like code.
- 04 **Scope autonomy by evidence.** Publish which task classes agents may do alone, with which checks. Review the list every quarter.
- 05 **Sandbox by default.** No production credentials in agent sessions. Apply the Rule of Two.
- 06 **Treat agent configuration as supply chain.** Rules files, skills, MCP servers and plugins get the same scrutiny as dependencies.
- 07 **Budget AI like cloud.** Workload owners, unit costs, limits and a monthly review.
- 08 **Stay portable.** Keep your context, rules and evaluation sets independent of any one vendor. Loyalty in this market is thin.
- 09 **Protect the first rung.** Keep hiring juniors, redesign their work, and test understanding without the tool.
- 10 **Build your own evaluation set.** Real tasks, kept private, re-run on every model change.

*Anyone can generate it.
Few can prove it. Be one of the few.*

METHOD

How this book was *made*

This book is a synthesis of public evidence, gathered and checked between September and 12 October 2026. It is not a systematic review, a benchmark or a replication. I conducted no interviews and ran no experiments of my own.

How sources were chosen

Primary sources came first: research papers, official reports, company announcements, documentation and earnings statements. Press coverage is used only where no primary was available, and is marked. Every number in the book was re-checked against its source, or a named relay where a page was blocked, on 8 October 2026, with four refreshed on 12 October 2026, and several figures from earlier drafts were corrected or dropped as a result.

Evidence classes

● PRIMARY STUDY	A measured result with a stated method: trials, field studies, payroll and postings data, peer-reviewed or preprint research.
● SURVEY	What respondents report about themselves. Useful for adoption and sentiment, weak for outcomes.
● VENDOR	A company describing its own product, data or results, including earnings calls and vendor-run research.
● ANALYST	A forecast. Labelled as such.
● OUR CONCEPT	Frameworks, scenarios and views that belong to this book. These are proposals to test. Examples marked illustrative show the shape of an idea, not real values. Where a mark says "our view", it means the author's.

What I left out, and why

- Revenue figures repeated in the press but not confirmed by the companies.
- A widely quoted analyst claim about ISO/IEC 42001 adoption with no primary source.
- The Stack Overflow 2026 trust figure. Its question changed, so it cannot be compared with 2025.
- Unattributed accuracy figures for Jev that could not be traced to a primary.
- Real-world "slopsquatting" attacks. Package hallucination is proven; I found no confirmed attack at scale.

Limits to keep in mind

Much of the best data in this field comes from companies that sell AI or AI-related tools. I name them so you can judge for yourself. Studies date quickly; the year of each is given, and several describe tools that have since been replaced. Surveys over-represent enthusiasts. Company percentages for "AI-written code" are not defined the same way and should not be compared directly.

Independence

This is an independent point of view. It was not commissioned or reviewed by any company named in it, and company names are used only to identify sources.

Edition

Edition 01, version 1.1, 12 October 2026. Version 1.0 was published on 8 October 2026; version 1.1 refreshes the METR, Veracode, Claude Code and New York Fed figures with newer sources.

SOURCES

Where the numbers *come from*

All sources accessed on or before 12 October 2026. Where a company page was blocked to automated access, a reliable relay is named. Web addresses are shown without the protocol.

PART I · CODE GOT CHEAP

- 01 N. Friedman. Introducing GitHub Copilot: your AI pair programmer. GitHub Blog, 29 Jun 2021. github.blog/news-insights/product-news/introducing-github-copilot-ai-pair-programmer
- 02 E. S. and B. Zhang. Building effective AI agents. Anthropic Engineering, 19 Dec 2024. anthropic.com/engineering/building-effective-agents
- 03 R. Robbes, T. Matricon, T. Degueule, A. Hora, S. Zacchioli. Agentic Much? Adoption of Coding Agents on GitHub. ACM TOSEM, 2026. [arXiv 2601.18341v2](https://arxiv.org/abs/2601.18341v2). doi.org/10.1145/3822180
- 04 TypeSafe AI. Introducing System One Models & Jev. D. Almeida, 15 Sep 2026. typesafe.ai/blog/introducing-system-one-models-and-jev
- 05 Microsoft. FY2026 Q4 earnings call, 29 Jul 2026 (Copilot users; agent share of pull requests). microsoft.com/investor
- 06 Anthropic. Anthropic acquires Bun as Claude Code reaches \$1B milestone, 3 Dec 2025. anthropic.com/news
- 07 CNBC. SpaceX to acquire the AI coding startup Cursor for \$60 billion, 16 Jun 2026. cnbc.com/2026/06/16/spacex-spcx-cursor-acquisition-ipo.html
- 08 OpenAI. Codex is becoming a productivity tool for everyone, 2 Jun 2026. openai.com/index/codex-for-knowledge-work, via Constellation Research
- 09 Alphabet. Q2 2026 earnings remarks, 22 Jul 2026; S. Pichai, Google Cloud Next 2026 keynote, 22 Apr 2026. blog.google
- 10 Cognition. Crosses \$1B in annualized revenue run rate, 25 Sep 2026; AI Productivity Guarantee, 4 Jun 2026. cognition.com/blog
- 11 AWS. Amazon Q Developer end-of-support announcement, 30 Apr 2026; Kiro is generally available, 17 Nov 2025. kiro.dev/blog/general-availability
- 12 M. Bogdanov. AI Coding Agents: Adoption Trends. JetBrains Research, Aug 2026. blog.jetbrains.com/research/2026/08/ai-coding-agent-adoption-2026
- 13 A. Karpathy. Posts on vibe coding (2 Feb 2025), context engineering (Jun 2025) and agentic engineering (8 Feb 2026), via The New Stack and PureAI. thenewstack.io/vibe-coding-is-passe
- 14 GitHub. Spec Kit; Visual Studio Magazine, 12 May 2026. github.com/githubb/spec-kit
- 15 Anthropic. Donating the Model Context Protocol and establishing the Agentic AI Foundation, 9 Dec 2025; MCP 2026-07-28 specification. blog.modelcontextprotocol.io
- 16 Linux Foundation. Agent2Agent protocol project, 23 Jun 2025. AGENTS.md, accessed 8 Oct 2026. agents.md
- 17 Bloomberg Law. OpenAI's \$3 billion deal to buy Windsurf falls apart (Google licence and hires), 11 Jul 2025. news.bloomberglaw.com
- 18 TechCrunch. Microsoft CEO says up to 30% of the company's code was written by AI, 29 Apr 2025. techcrunch.com
- 19 METR. Measuring AI ability to complete long software tasks (arXiv 2503.14499v4); Time Horizon 1.1, 29 Jan 2026; time-horizons data, 8 May 2026. metr.org/time-horizons
- 22 P. Whitfill, C. Wu, J. Becker, N. Rush. Many SWE-bench-passing PRs would not be merged into main. METR, 10 Mar 2026; and METR research update, 12 Aug 2025. metr.org/notes/2026-03-10-many-swe-bench-passing-prs-would-not-be-merged-into-main
- 23 Sonar. Critical verification gap in AI coding (State of Code survey), 8 Jan 2026. sonarsource.com/company/press-releases
- 24 K. Z. Cui, M. Demirer, S. Jaffe, L. Musolff, S. Peng, T. Salz. The Effects of Generative AI on High-Skilled Work. Management Science, 2026. doi.org/10.1287/mnsc.2025.00535
- 25 J. Becker, N. Rush, E. Barnes, D. Rein. Measuring the impact of early-2025 AI on experienced open-source developer productivity. METR, 10 Jul 2025. [arXiv 2507.09089](https://arxiv.org/abs/2507.09089). metr.org/blog/2025-07-10-early-2025-ai-experienced-os-dev-study
- 26 METR. We are changing our developer productivity experiment design, 24 Feb 2026. metr.org/blog/2026-02-24-uplift-update
- 27 E. Murphy-Hill, J. Butler, A. Savelieva. Adoption and impact of command-line AI coding agents. [arXiv 2607.01418](https://arxiv.org/abs/2607.01418), 1 Jul 2026. arxiv.org/abs/2607.01418
- 28 S. Maier et al. A meta-analysis of the effect of generative AI on productivity and learning in programming. [arXiv 2605.04779](https://arxiv.org/abs/2605.04779), 6 May 2026. arxiv.org/abs/2605.04779
- 29 Faros AI. The AI Productivity Paradox, Jul 2025; AI Engineering Report 2026: The Acceleration Whiplash. faros.ai/research
- 30 Atlassian. State of Developer Experience Report 2025. atlassian.com/blog/developer/developer-experience-report-2025
- 31 N. Harvey, D. DeBellis. Announcing the 2025 DORA Report; K. Storer, D. DeBellis. DORA AI Capabilities Model. Google Cloud, 24 Sep 2025. cloud.google.com/blog/products/ai-machine-learning/announcing-the-2025-dora-report
- 32 Invariant Labs. GitHub MCP exploited, 26 May 2025. invariantlabs.ai/blog/mcp-github-vulnerability
- 33 AWS Security Bulletin AWS-2025-015 (CVE-2025-8217), 23 Jul 2025. aws.amazon.com/security/security-bulletins/AWS-2025-015
- 34 Fortune. AI coding tool Replit wiped database, 23 Jul 2025. fortune.com
- 35 Wiz Research. s1ngularity: supply chain attack leaks secrets on GitHub, Aug 2025. wiz.io/blog/singularity-supply-chain-attack
- 36 Snky. ToxicSkills, 5 Feb 2026; Koi Security via The Hacker News, Feb 2026. snky.io/blog/toxicskills-malicious-ai-agent-skills-clawhub
- 37 M. Nasr et al. The attacker moves second. [arXiv 2510.09023](https://arxiv.org/abs/2510.09023), Oct 2025. arxiv.org/abs/2510.09023
- 38 NIST CAISI. Strengthening AI agent hijacking evaluations, 17 Jan 2025. nist.gov
- 39 Anthropic Engineering. Making Claude Code more secure and autonomous with sandboxing, 20 Oct 2025. anthropic.com/engineering/claude-code-sandboxing
- 40 GitGuardian. The State of Secrets Sprawl 2026, Mar 2026. gitguardian.com/state-of-secrets-sprawl-report-2026
- 41 TechCrunch. OpenAI says AI browsers may always be vulnerable to prompt injection, 22 Dec 2025. techcrunch.com
- 42 Meta AI. Agents Rule of Two, 31 Oct 2025. ai.meta.com/blog/practical-ai-agent-security
- 43 European Commission. Cyber Resilience Act; AI Act; Council of the EU, 29 Jun 2026. OMB M-26-05, 23 Jan 2026. digital-strategy.ec.europa.eu

PART II · TRUST GOT HARDER

- 20 Fortune. Google says a quarter of new code is AI-generated (Alphabet Q3 2024 call), 30 Oct 2024; Business Insider, Pichai on engineering velocity, Jun 2025. fortune.com/2024/10/30/googles-code-ai-sundar-pichai
- 21 Veracode. 2025 GenAI Code Security Report, 30 Jul 2025; Spring 2026 update; 2026 GenAI Code Security Report, 28 Jul 2026. veracode.com/blog/2026-genai-code-security-report-ai-risk

SOURCES CONTINUED

PART III · NEW ECONOMICS, NEW BUILDING BLOCKS

- 44 B. Cottier et al. LLM inference prices have fallen rapidly but unequally across tasks. Epoch AI, 12 Mar 2025. epoch.ai/data-insights/llm-inference-price-trends
- 45 Anthropic. Claude Code docs: Manage costs effectively, accessed 8 Oct 2026. code.claude.com/docs/en/costs
- 46 Jellyfish. AI Engineering Trends report, 20 May 2026, via DevOps Digest. devopsdigest.com
- 47 M. Rodriguez. GitHub Copilot is moving to usage-based billing. GitHub Blog, 27 Apr 2026. github.blog/news-insights/company-news/github-copilot-is-moving-to-usage-based-billing
- 48 OpenAI. Introducing SWE-bench Verified, 13 Aug 2024; Why we no longer evaluate SWE-bench Verified, Feb 2026 (read via the r.jina.ai reader; date from secondary sources). openai.com/index/why-we-no-longer-evaluate-swe-bench-verified
- 49 X. Deng et al. SWE-Bench Pro. arXiv 2509.16941; Scale AI public leaderboard, accessed 8 Oct 2026. labs.scale.com/leaderboard/swe_bench_pro_public
- 50 Anthropic Engineering. Demystifying evals for AI agents, 9 Jan 2026. anthropic.com/engineering/demystifying-evals-for-ai-agents
- 51 TypeSafe AI documentation: Introduction, System One, AI primer, Confidence, Models, API, Coding agents, Jev 1.13 jaggedness. Accessed 8 Oct 2026. docs.typesafe.ai
- 52 R. Guo, Y. Liu, G. Deng et al. Just Ask Jev: RLCD as a zero-shot detector of AI alignment failures. arXiv 2609.29429, 24 Sep 2026. arxiv.org/abs/2609.29429
- 53 C. Guo, G. Pleiss, Y. Sun, K. Q. Weinberger. On calibration of modern neural networks. ICML 2017, PMLR 70. proceedings.mlr.press/v70/guo17a.html

PART IV · VERIFIED DELEGATION

- 54 M. Souppaya, K. Scarfone, D. Dodson. Secure Software Development Framework v1.1. NIST SP 800-218, Feb 2022; SP 800-218A, Jul 2024. csrc.nist.gov/pubs/sp/800/218/final
- 55 S. Bursuc et al. A benchmark for vericoding: formally verified program synthesis. arXiv 2509.22908, 2025. arxiv.org/abs/2509.22908
- 56 M. Maaz et al. Agentic property-based testing. arXiv 2510.09907, 2025. arxiv.org/abs/2510.09907
- 57 M. Harman et al. Mutation-guided LLM-based test generation at Meta. FSE 2025. arXiv 2501.12862. arxiv.org/abs/2501.12862
- 58 R. Gopu, D. Apirian. 60 million Copilot code reviews and counting. GitHub Blog, 5 Mar 2026. github.blog/ai-and-ml/github-copilot/60-million-copilot-code-reviews-and-counting

PART V · PEOPLE AND THE ROAD AHEAD

- 59 E. Brynjolfsson, B. Chandar, R. Chen. Canaries in the coal mine? Stanford Digital Economy Lab, Nov 2025; project page, Aug 2026. digitaleconomy.stanford.edu/publications/canaries-in-the-coal-mine
- 60 Indeed Hiring Lab. US labor market snapshot, 14 May 2026; FRED series IHLIDXUSTPSOFTDEVE. fred.stlouisfed.org/series/IHLIDXUSTPSOFTDEVE
- 61 Federal Reserve Bank of New York. The labor market for recent college graduates, Feb and Aug 2026. newyorkfed.org/research/college-labor-market
- 62 J. H. Shen, A. Tamkin. How AI assistance impacts the formation of coding skills. Anthropic, 29 Jan 2026. anthropic.com/research/AI-assistance-coding-skills
- 63 Fortune. IBM tripling entry-level hiring, 13 Feb 2026. The Register. AWS CEO on junior staff, 21 Aug 2025. fortune.com

— GLOSSARY

Words this book *uses*

Agent

A model with tools running in a loop towards a goal, choosing its own steps: read, edit, run, retry.

Agentic workflow

A system that wires model calls, tools and checks together along a path written in code.

AUROC

How well a score ranks positive cases above negative ones across every threshold. 0.5 is chance, 1.0 is perfect. It is not an accuracy rate.

Calibration

Whether stated probabilities match reality. Of answers given at 80%, about 80% should be right. True of groups of answers, never a promise about one.

Context engineering

Choosing exactly what information a model sees for the next step, and what it must not see.

Cost per accepted change

All model, infrastructure and human cost divided by the changes that met the agreed quality bar. The unit this book recommends.

Evaluation harness

The tasks, graders and environment used to test a model or agent on your own work, repeatably.

MCP

Model Context Protocol. A standard way for agents to connect to tools and data. Now governed by the Linux Foundation's Agentic AI Foundation.

pass@k and pass^k

pass@k: the agent succeeds at least once in k tries. pass^k: it succeeds every time. Production needs the second.

Prompt injection

Instructions hidden in content an agent reads, such as an issue, a web page or a file, that hijack what it does.

RLHF, RLVR, RLCD

Three ways to train a model after pretraining: rewarding text people prefer, answers a program can verify, or decisions with honest probabilities (TypeSafe's method).

Rule of Two

Meta's guideline that an agent session should combine at most two of: untrusted input, sensitive access, and the power to act externally.

Sandbox

An isolated environment, with limits on files and network, where an agent can work without touching production.

Spec-driven development

Writing requirements, design and tasks into the repository before any code, so agents and people build against the same definition of done.

System One model

TypeSafe AI's term, after Kahneman, for a model that returns fast typed decisions with probabilities instead of generating text.

Time horizon

METR's measure: the length of task, in human expert time, an AI completes with a given reliability, usually 50% or 80%.

Typed output

An answer constrained to options defined in advance. It guarantees the form of the answer, not its correctness.

Verified delegation

This book's proposal: widen what agents may do only where the evidence to accept their work exists.

Vibe coding

Karpathy's 2025 term for building by prompting and accepting output without reading the code. Fine for weekend projects.



ABOUT THE AUTHOR

Sanjeev Azad

Advisor, AI strategist, author and innovator



Sanjeev Azad helps organisations move from AI experimentation to production-scale impact. He leads enterprise AI strategy and innovation for a global engineering organisation, working across APAC and beyond.

He began as a software engineer, moved into architecture in financial services, and then spent years as a technology architect on modernisation programmes for banking, capital markets and automotive clients across the UK, Europe and the US. He has since led a regional architecture practice, served as a regional CTO, and now leads global innovation.

His work centres on enterprise AI transformation, AI operating models, AI-native engineering, agentic frameworks and AI governance in regulated businesses. He works with regulators, industry bodies and executives on what it takes to adopt AI well.

He also builds open frameworks and tools, most of them free and MIT-licensed, including The Value Cube, The Enterprise EPIC, Where the Gain Goes and the SAE Framework.

*Code is cheap. Trust isn't.
Anyone can generate it. Few can prove it.*



SCAN TO MEET THE AUTHOR

sanjeev-azad.pages.dev/about