

What makes AI coding agents effective

New research from Span on what improves cost, autonomy and rework



Stephen Poletto

Field CTO, Span

Adoption was the game in 2025. Effectiveness is the game in 2026.

Adoption

Are engineers using it?

How many seats, how many daily actives?

Which tool, which model?

2025

Effectiveness

What is it producing?

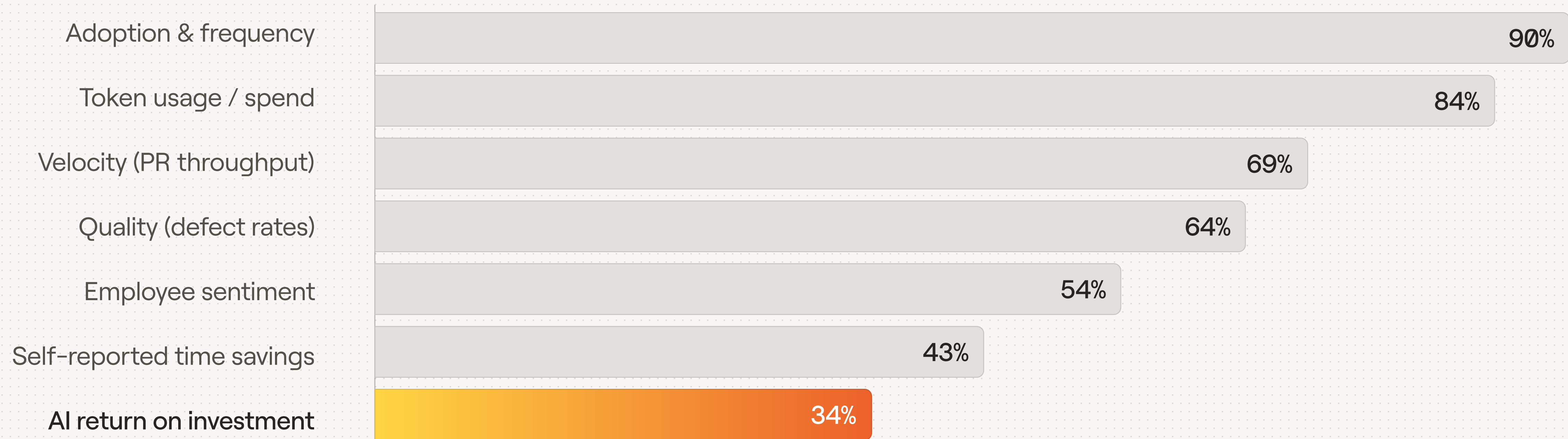
How do we increase the yield per dollar?

Who is using it effectively vs. superficially?

How do we leverage human attention?

2026

Most teams measure usage, not outcomes



More leaders track how engineers feel about AI than whether it pays off. Only 24% track velocity, quality and ROI together.

01

State of the industry

Where engineering orgs are on their AI journey

100+

engineering leaders surveyed,
Q3 2026

~3 in 4

are directors, VPs or CTOs

~50%

lead orgs of 200+ engineers

25%+

lead orgs of 500+ engineers

AI now writes most of the code

70%

median share of last month's merged code written with AI assistance

73%

say at least half their merged code is AI-assisted

Work is shifting from authoring to delegation

AI-assisted authoring

AI completes lines and answers questions

96%

Interactive agent

Agent does multi-step work; developer steers live

93%

Reviewed delegation

Agent returns a PR; a human reviews every one

89%

Trusted delegation

Agent PRs merge with sampled or no review

38%

Agents are leaving the IDE

79%

let agents pick up work straight from tooling: tickets (65%), PR events (63%)

46%

say a typical developer runs 2+ agents in parallel

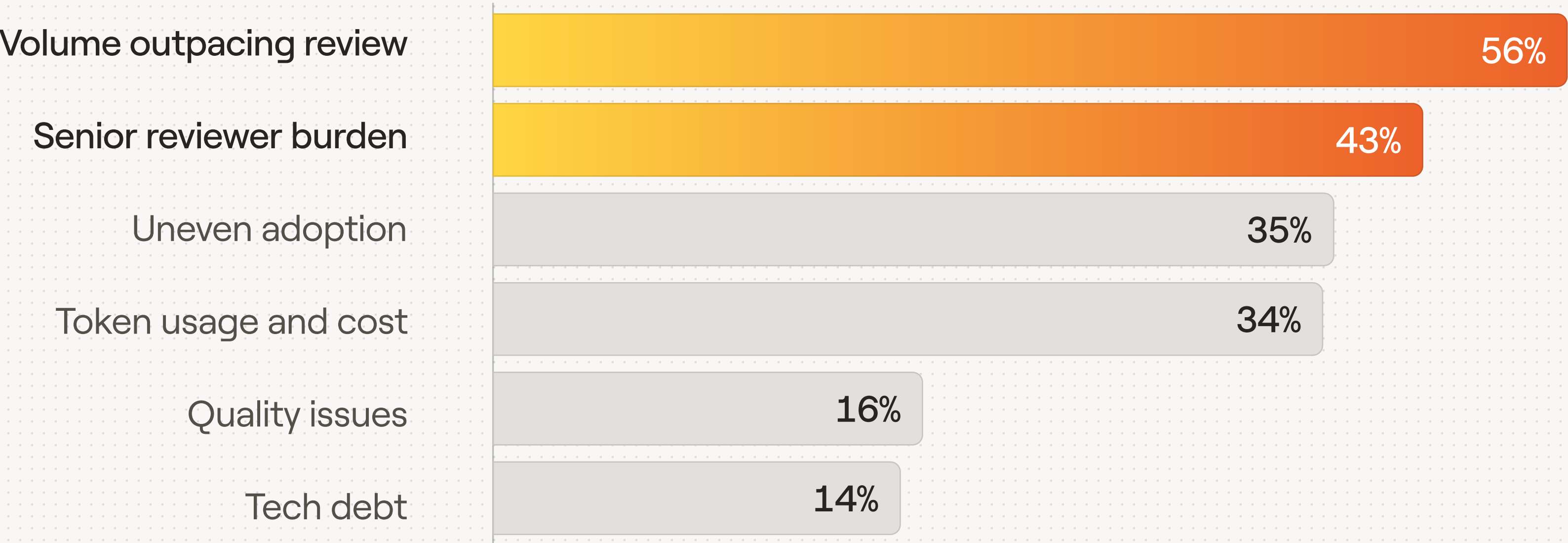
38%

trigger agents from automated signals with no human in the loop

66%

still run agent workloads mostly or entirely on local machines. The work is going to the background faster than the infrastructure is.

Review capacity is the most acute problem



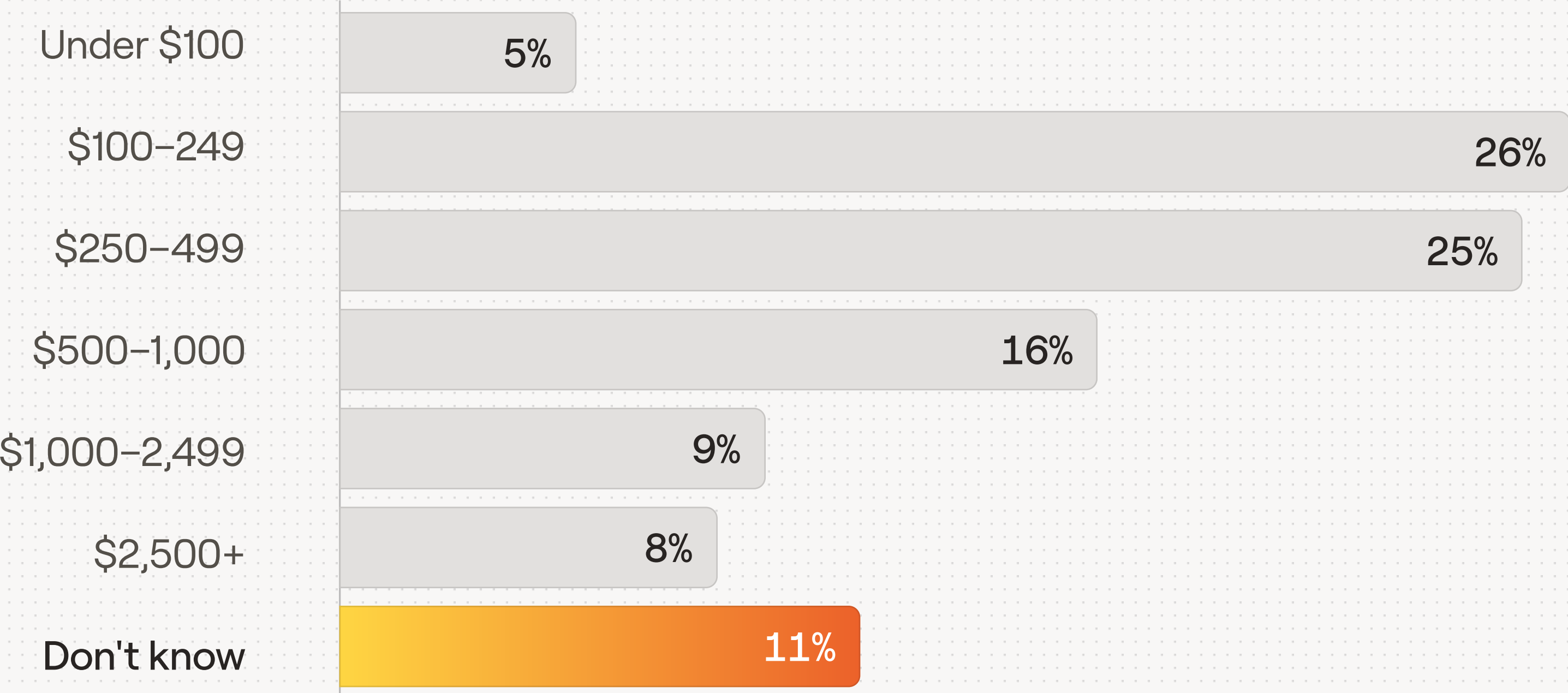
73%

Teams running 2+ agents in parallel: **68%** cite review volume, vs. **47%** of those running one at a time.

picked review volume, senior-reviewer burden, or both as a top-two problem

Cost management is an under-developed discipline

MONTHLY AI CODING SPEND PER DEVELOPER



83%

have no firm limit on AI tooling spend

43%

say over a quarter of their devs just use the default model

Source: Span survey, Q3 2026.

The gap between top and bottom performers is widening

68%

say the gap between their strongest and weakest engineers has widened since adopting AI

26%

estimate an output gap of 5x or more on similar work

Effective practice lives with individuals and rarely becomes a team standard

Spread informally

A practice reaches a few nearby teammates or travels through a champion, and nothing is captured.

78%

Become a standard

Fewer than one in five say effective practices get captured and rolled out as a team or org standard.

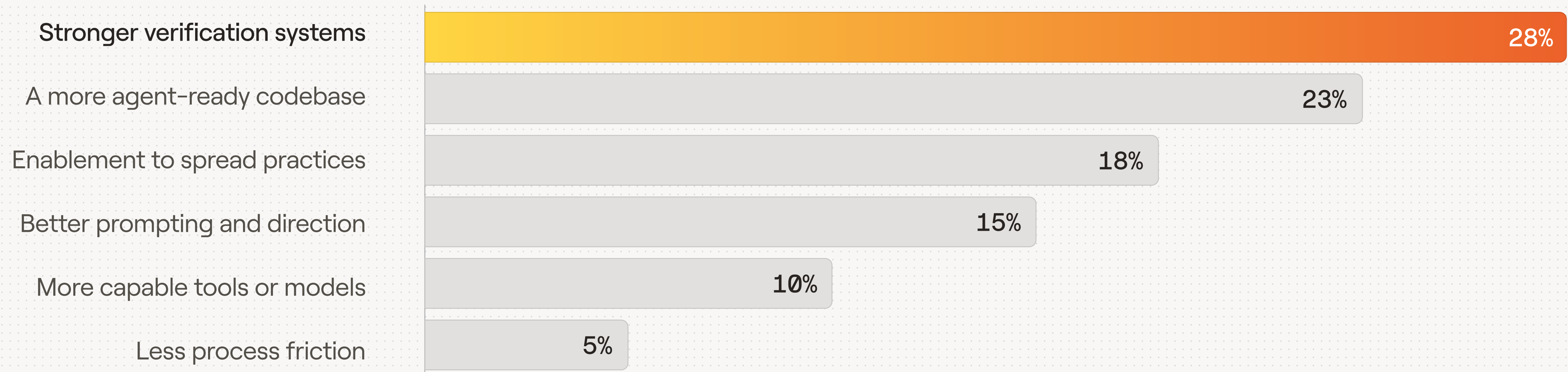
19%

Maintain reusable artifacts

Only a third say their best engineers keep reusable prompts, rules files or agent configs: the things that let a practice spread.

33%

Leaders want better systems, not better models



Among leaders who cite review volume as a top problem, **40%** pick stronger verification, vs. **11%** of everyone else.

Key trends

01

Work goes agentic

Delegation is becoming more common; agents are triggered by tickets, PRs and alerts.

02

Cost has no owner yet

Spend is tracked, rarely tied to outcomes, and mostly uncapped.

03

The floor needs raising

The gap is widening, and effective practices stay local to top performers.

04

Review is the bottleneck

Code is cheap to write. Review and verification is undergoing reinvention.

02

Beyond the model

What distinguishes effective agent trajectories, measured from the trajectories themselves

100+

engineering teams

12 weeks

of real work, May–July 2026

Full traces

prompt to merge or abandonment

Effective trajectories spend fewer tokens and less human attention

LOW COST

Token cost per merged AI line

Total spend in the trajectory ÷
AI-authored lines that survived
to merge

HIGH AUTONOMY

Turn yield

Merged AI-authored lines per
human turn

LOW REWORK

Review cycles per 1K AI lines

Requested-changes rounds per
1,000 merged AI lines

All three are lagging: you learn them after the tokens are spent.

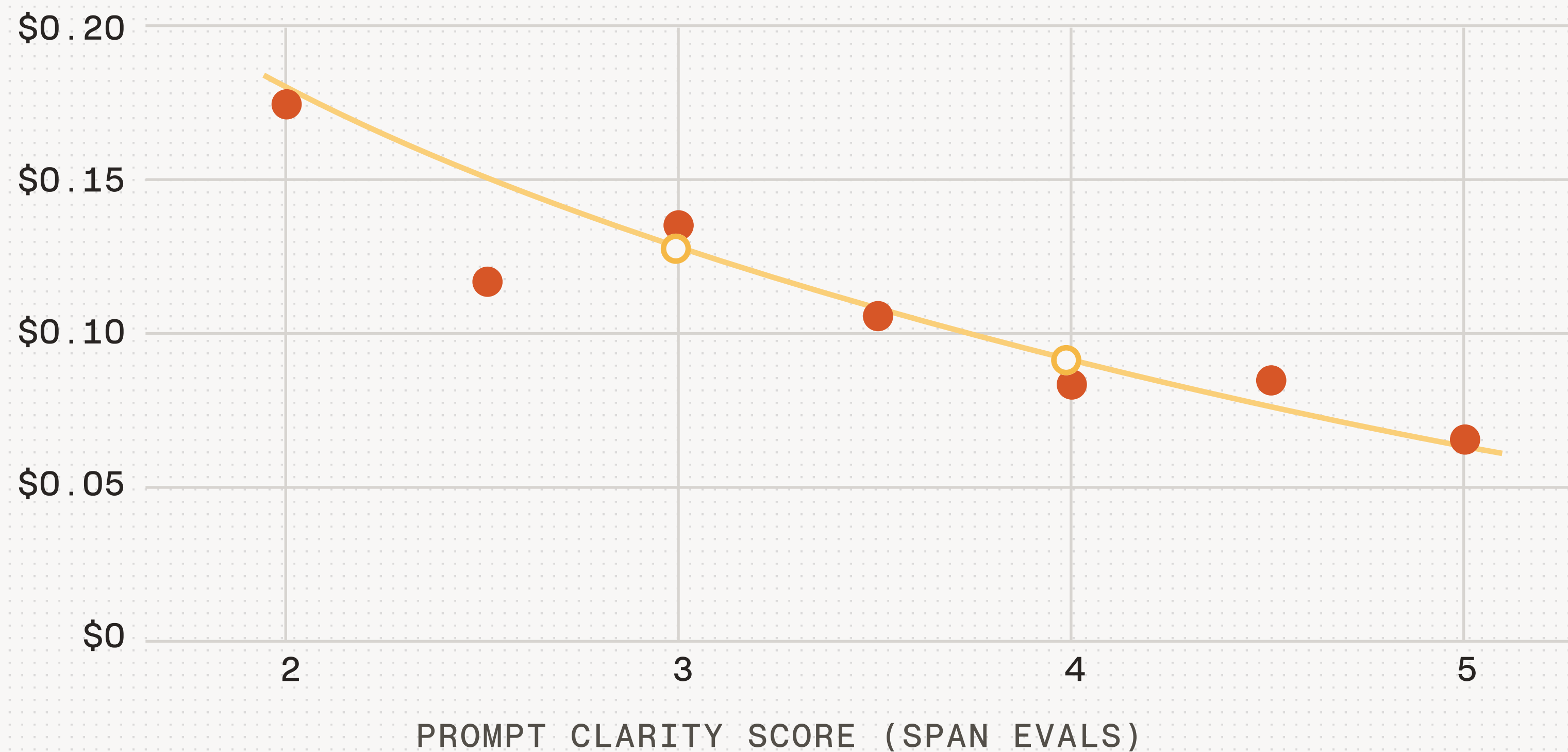
Three controllable predictors

Lever (Span eval, 1–5)	What it plainly means	Outcome	Per +1 point
Prompt clarity	The task is specified clearly	Cost per merged line	-27%
Environment readiness	The repo and tooling let the agent work and self-check	Turn yield	+88%
Quality stewardship	The developer steers toward standards and a safe rollout	Review cycles	-39%

Observational associations, not causal. Source: Span, Beyond the Model (Q3 2026).

Clearer prompts, cheaper code

TOKEN COST PER MERGED AI LINE



-27%

cost per merged AI line for each point of prompt clarity

$R^2 = 0.878$. Median PR (50 AI lines):
\$6.43 at clarity 3.0, \$4.72 at 4.0.

Write prompts like tickets

```
// team template: bug fix
```

Context issue, error, exemplar

Goal what should change

Done when tests X pass, Y still works

Constraints conventions, perf, security

Out of scope public API, deps

State what "done" means

The biggest token sink we saw: agents inferring, and mis-inferring, the definition of done.

Give pointers, not archaeology

"Follow the pattern in payments/refunds.ts." Every fact the agent rediscovers is spend.

Resolve ambiguity first

A two-minute clarification chat is cheaper than a wrong implementation.

Ready environments, more autonomous agents

MODELED TURN YIELD: MERGED AI LINES PER HUMAN TURN

Readiness 2.5

~5 lines

Readiness 4.5

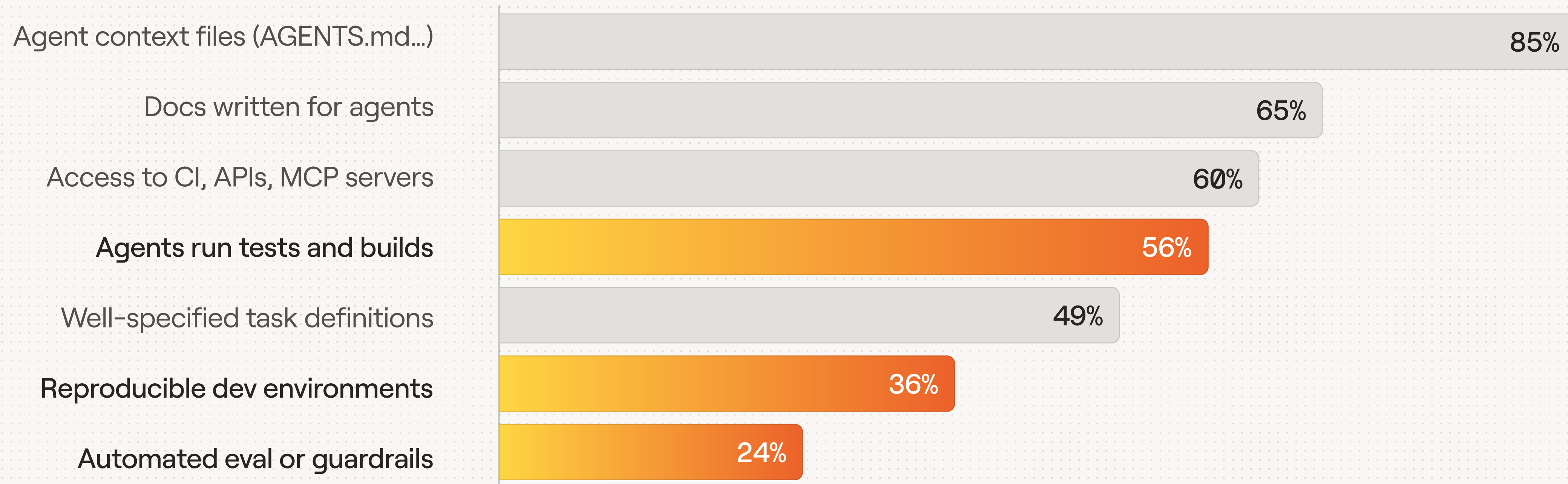
~17 lines

+88%

merged AI lines per human turn
for each point of readiness

$R^2 = 0.949$

Teams tell agents what to know, but rarely let them check



"What has your team put in place to make its environment work well for AI agents?" Source: Span survey, Q3 2026

Onboard the agent like a new hire

One-command setup

Install, build and test with one documented command.

Fast, non-flaky tests

Self-checking is the engine of autonomy. Fix flaky suites first.

Machine-readable feedback

Quick type checks, linters and builds let the agent iterate cheaply.

Script the hidden state

Seed data, stubs and mock services to let the agent self-run.

THE DRILL

Run an agent through a representative task from a fresh environment.

Log every time a human steps in, and why. Fix the top blocker. Repeat.

Stewardship up front, fewer review cycles later

MOST REVIEW CHURN IS THIS WORK, ARRIVING LATE

“ This doesn't match how we do it in this module.”

“ What happens when this is null?”

“ How do we roll this back?”

“ Where's the logging?”

–39%

review cycles per point of
stewardship: ~7 → ~4–5 per 1K AI
lines, from 3.0
to 4.0

$R^2 = 0.879$

Don't outsource the first review to your colleagues

BEFORE THE PR

Read the diff yourself

Check for scope creep, deleted tests and suspicious rewrites.

IN THE PROMPT

Make the agent prove it

Run the suite, exercise the path, then ask it to list what it couldn't verify.

FOR PRODUCTION

Surface blast radius

What does it reach, and how do we undo it? Ship gates and observability with it.

THE TAKEAWAY

Better models lift everyone.
Your edge is context quality,
environments and stewardship.

03

In the field

How leading teams are getting more effective

Uber

cost as an engineering problem

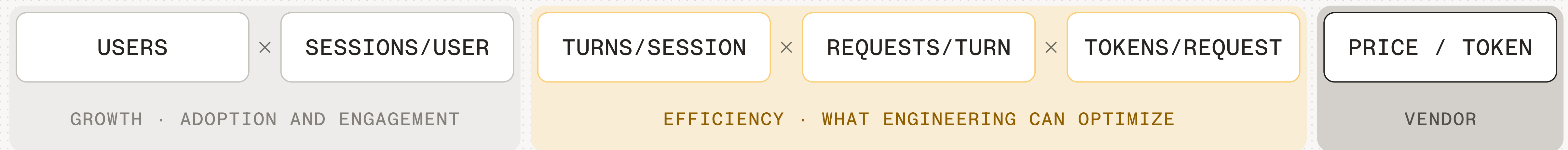
Fin

verification instead of inspection

Field notes

patterns across Span customers

Uber decomposed AI spend into an equation



7x

weekly active users

9.4x

weekly agentic requests

70%+

of PRs attributed to agents

Flat

total AI spend since April

Uber cut cost per session in half

Tune defaults

- Auto-compact at 400K tokens, even on 1M-context models
- Medium reasoning effort by default
- Cheaper models for well-defined subagent tasks
- 1-hour cache TTL for interactive sessions

Engineer context

- 1,000+ MCP tools exposed as CLI commands, loaded on demand: 50–70K tokens saved per session
- Context graph across 30+ systems: one query took 38 seconds with it, 20+ minutes (and a wrong answer) without

Make cost visible

- Live spend in every developer's status line
- Alerts at 50, 80 and 100% of expected spend
- Session analysis flags 16 anti-patterns, each with its dollar impact

–52%

cost per session from
June peak

–34%

cost per 1K requests,
same model

A council of reviewers at Fin

REVIEW DECOMPOSED INTO INDEPENDENT SUB-AGENTS

01 Is the problem description sound?

02 Does the diff match the intent?

03 Is it safe?

04 Is the logic correct?

05 Does it follow best practices?

93%+

of PRs agent-driven across two main codebases

19%+

auto-approved with no human reviewer

Auditor approved. Meets SOC 2, HIPAA, ISO 27001, ISO 42001 and AIUC-1.

What engineering leaders are telling us

SHARED INFRASTRUCTURE

“We’re building our own sandbox environments and skills and plugins... to help our engineers be more effective.”

SVP of Engineering, late-stage collaboration software company

COST

“This constant campaign of... you don’t need to use the top model for literally everything. Have you heard about pinning models to skills?”

Director of Engineering, Fortune 500 e-commerce company

AUTONOMY

“For low-risk PRs, it’ll auto-approve... it runs multiple reviewers, not just one cheap model.”

Co-founder & CTO, Series B software company

04

Code review becomes verification engineering

Mandatory human review of every diff won't survive long

For 30 years, code review assumed humans write code slower than other humans can read it.

Higher up the abstraction ladder we go

Machine code
1950s

Assembly
Hand-tuned, fully
inspected

Compilers
FORTRAN, 1957:
“reckless to trust”

Source code
Today: where humans
inspect every change

Spec and intent
Next: where review
moves

Four shifts already underway

01

Verification replaces inspection

Fin, Rewind: councils of specialist agents

Ona: runs code in a real environment

StrongDM: mocks out "digital twins"

02

Risk routes human attention

Meta: 10% of diffs flagged, 60% of incidents caught

Rootly: AI is sole approver for low-risk changes

03

Review moves upstream to the spec

A vague spec yields a hundred confident, wrong PRs

Review the spec, not the diff

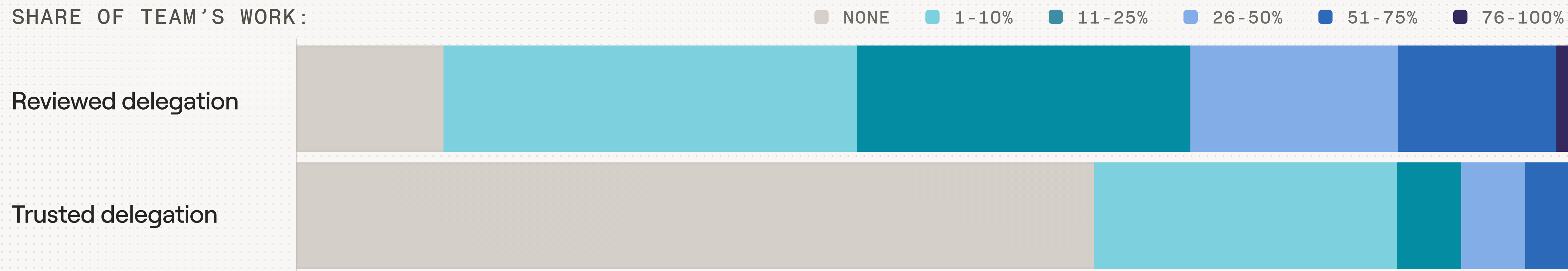
04

Compliance controls are being reworked

Rewind, Ona, Fin: have all reworked their compliance controls to allow for agentic review

38% already let some agent PRs merge without full review

SHARE OF TEAM'S WORK:



89%

do some reviewed delegation: an agent returns a PR, a human reviews it.

14%

already put more than a tenth of their work through trusted delegation. That tail is the frontier.

The three levers are verification engineering at small scale

Prompt clarity



Spec review. The cheapest place to catch a flaw.

Environment readiness



Self-verification. Agents prove their own work.

Quality stewardship



Evidence ships with the change. Risk is surfaced before merge.

My predictions for 2028

Most diffs go unread by humans: changes merge on verification alone

Human review is a routing outcome: reserved for trust boundaries and irreversible ops

Spec review is the main quality gate: flaws are cheapest to catch before code exists

Diff viewers give way to attention routers: tools say where to look, and where not to

Safety comes from verification: risk scores, gates, staged rollout, fast rollback

Compliance catches up: the automated audit trail is richer than a click

Lessons from the frontier

MEASURE

Track cost per merged line, turn yield and review cycles on AI-generated code

CLARIFY

Ship prompt templates for your top three task types

READY

Run an agent from a fresh environment; fix the top blocker; repeat

SPREAD

Capture one top performer's workflow and make it a team standard

EXPOSE COST

Put spend feedback in front of developers, and tie budget increases to an expected result

VERIFY

Risk-tier your PRs; automate verification for one low-risk class, with auditors in early

Questions?

The leverage is beyond the model: in how you shape your harness and team practices



Stephen Poletto

Field CTO, Span

stephen@span.app

Get the full research:

span.app