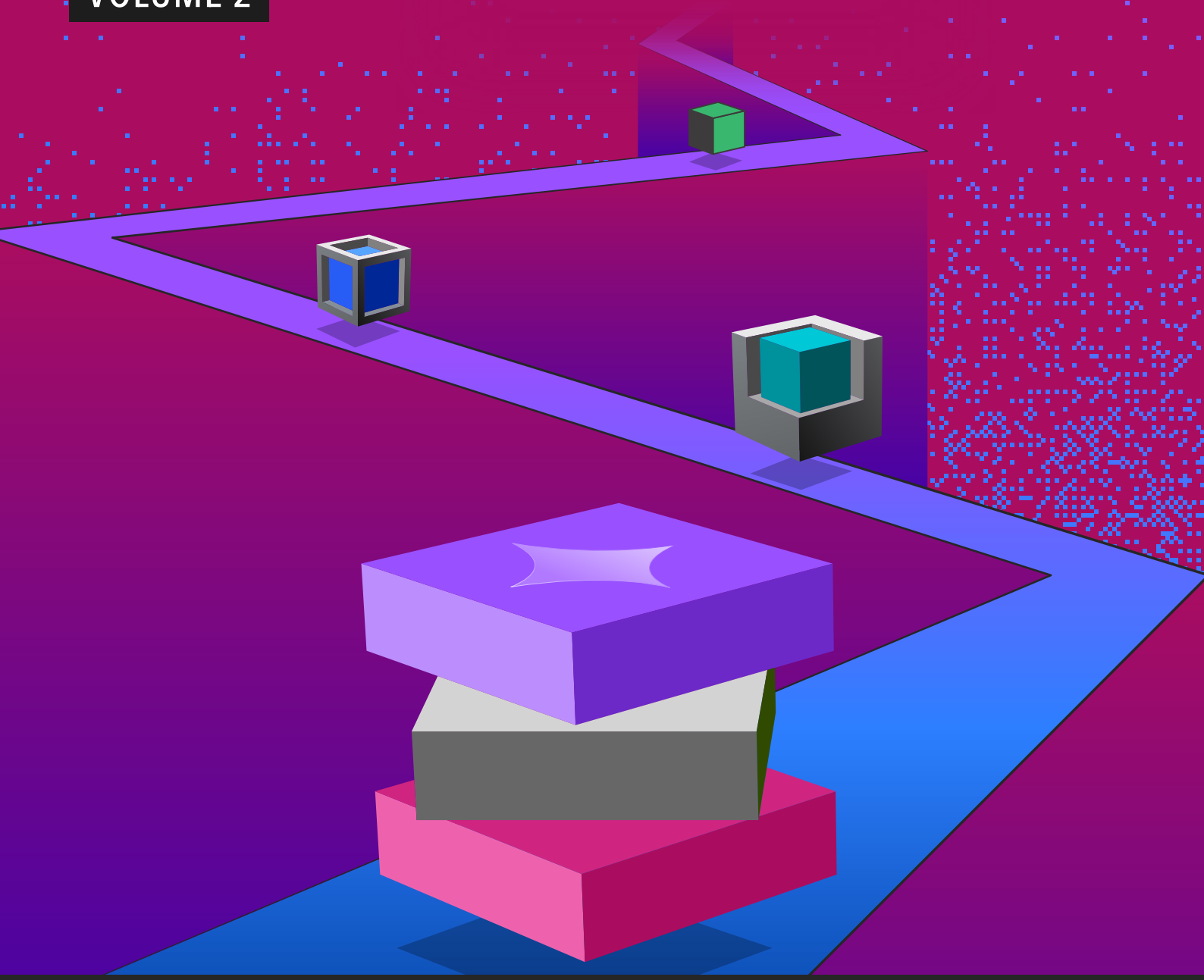
Weave Intelligence

State *of* AI in Platform Engineering

VOLUME 2



WEAVE INTELLIGENCE PRESENTS
STATE OF AI IN PLATFORM ENGINEERING VOLUME 2
PUBLISHED IN 2026

State *of* AI in Platform Engineering Volume 2

Table of contents

Authors

Sam Barlien

RESEARCHER, WEAVE INTELLIGENCE

Sam Barlien is a researcher at Weave Intelligence, the research arm of platformengineering.org, the world's largest platform engineering community. With more than 10 years tracking technology communities and ecosystems, he brings first-hand perspective to his research on platform engineering and industry trends. He contributes to Weave Intelligence reports and conducts the weekly research interview series. He co-hosts PlatformCon, the world's largest platform engineering conference, and contributes to Platform Weekly, read by 100,000 engineers every week.

Luca Galante

LEAD ANALYST, WEAVE INTELLIGENCE

Luca Galante is the Core Contributor to the Platform Engineering community, the world's largest platform engineering community with over 200,000 members. He routinely speaks to dozens of engineering teams every month, and summarizes his learnings and takeaways from hundreds of setups into crisp, insightful content for everyone in the industry, from beginner-Ops to cloud experts. He is the host of PlatformCon, the world's largest platform engineering event, and writes to over 100,000 engineers every Friday in his newsletter, Platform Weekly.

Florian Lipp

SENIOR ANALYST, WEAVE INTELLIGENCE

Dr. Florian Lipp is an analyst at Weave Intelligence, where he researches how platform engineering teams design, operate, and evolve their platforms. His current work spans topics including cloud economics and observability, helping teams move beyond dashboards and cost reports toward feedback systems that actually inform decisions. He co-organizes PlatformCon, curates its content program, and serves as editor in chief at platformengineering.org, two of the community's most important gathering points.

About *our* partners

Thoughtworks is a global technology consultancy that integrates design, engineering and AI to drive digital innovation. We are over 10,000 people strong across 47 offices in 18 countries. For 30+ years, we've delivered extraordinary impact together with our clients by helping them solve complex business problems with technology and culture as the differentiator. Visit Thoughtworks.

Vultr is on a mission to make high-performance cloud infrastructure easy to use, affordable, and locally accessible for enterprises and AI innovators around the world. Vultr is trusted by hundreds of thousands of active customers across 185 countries for its flexible, scalable, global Cloud Compute, Cloud GPU, Bare Metal, and Cloud Storage solutions. Founded by David Aninowsky and self-funded for over a decade, Vultr has grown to become the world's largest privately held cloud infrastructure company. Learn more at Vultr.

Syntasso builds platform orchestration technology designed for a world where both humans and AI agents consume enterprise platforms.

Syntasso Kratix Enterprise (SKE) is an AI-native platform orchestrator that turns an organisation's existing infrastructure, automation, policies, and workflows into governed, reusable platform capabilities. Platform teams define these capabilities once and make them available through the interfaces their consumers need, from developer portals, APIs and CLIs to MCP and AI agents, while consistently managing lifecycle, policy, and operations across environments and fleets.

Syntasso's approach is to make the platform itself the control boundary, so the same capabilities, policies and workflows can safely serve human and machine consumers.

Syntasso Kratix Agentic (SKA) is a focused subset of SKE for organisations prioritising building AI agents into platform and software delivery workflows. It provides governed interfaces for agents to interact with enterprise technology, helping organisations adopt agentic ways of working without giving non-deterministic systems unrestricted access to the infrastructure beneath them.

Both SKE and SKA are built on Kratix, Syntasso's open source framework for

enterprise platform orchestration. The model is simple: platform teams and their agents produce reusable capabilities, developers and AI agents consume them through appropriate interfaces, and the platform maintains their lifecycle and governance across the enterprise.

env zero is the company behind the autonomous cloud control plane for large enterprises running infrastructure at scale. Formed through the merger of env zero and CloudQuery in March 2026, the platform pairs CloudQuery's ability to continuously extract every cloud and infrastructure source into a single central data store with env zero's IaC governance, then layers a knowledge graph on top, so the whole estate can be understood and operated deterministically by both people and AI.

The platform continuously discovers shadow resources, detects drift and compliance violations, and closes the loop from detection to verified fix, without requiring engineers to stitch together a multitude of tools to answer one question.

Leading enterprises including PayPal, Broadcom, Paramount, MongoDB, Walmart, Fidelity, Visa, Western Union, Pismo, Adaptavist, Automation Anywhere, and Virgin Media O2 use the platform to standardize and govern infrastructure delivery across teams, clouds, and IaC frameworks. Visit Envzero.

Microsoft helps developers and engineering teams build, modernize, and ship AI-powered software with confidence. Across Azure, GitHub, Visual Studio Code, and Microsoft Security, we bring together trusted cloud infrastructure, developer platforms, model choice, security, governance, and business context across the full software development lifecycle. Our focus is helping organizations move from AI experimentation to production-grade delivery, supporting teams as they plan, code, review, test, deploy, operate, and learn at scale.

Customers such use GitHub Copilot and GitHub Enterprise to improve developer experience, increase successful builds, accelerate pull requests, and help teams modernize code with more confidence.

Through AI Builders, our always-on digital series for developers, Microsoft shares practical guidance on how AI is reimagining the developer workflow.

Key takeaways

Shipping more is not the same as earning more

38% of organizations now ship at least twice as much, and 8% describe the return as transformative. The plateau we named in our last report has not disappeared. It moved downstream, from between adopting AI and delivering with it to between delivering and proving the delivery was worth anything.

Maturity pays at a threshold, not as a gradient

Deploying agents without rebuilding the platform underneath them moves throughput by six percentage points. Rebuilding the platform moves it by twenty-four. The whole return sits on one step of the climb, and most of the industry is standing below it.

The industry names the platform as its own constraint

Asked what most obstructs scaling AI, respondents put platform readiness first at 27%, ahead of cost at 23% and review bandwidth at 16%. Model capability came fourth at 11%. Nobody is waiting for a better model.

The bottleneck moved off writing code and into absorbing it

A third of organizations still validate AI-generated work by human reading alone, and 17% run automated validation loops. Generation industrialized. Absorption did not.

Most organizations cannot attribute what their agents do

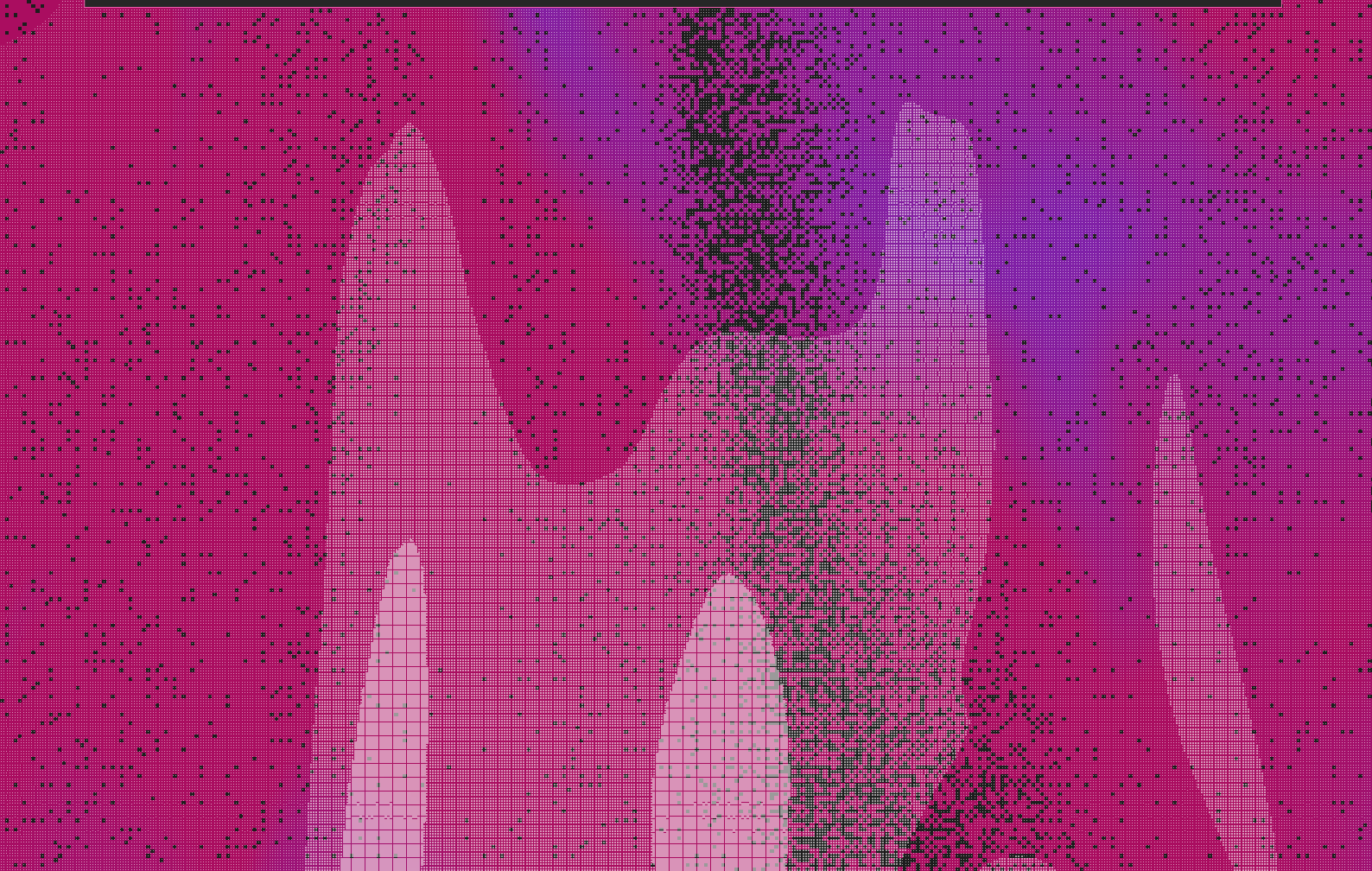
A quarter run agents on borrowed human credentials, another 12% on shared service accounts, and 18% of respondents cannot say how agent identity is managed. An agent that cannot be identified cannot be audited, scoped, or switched off without switching off a person, and every control downstream inherits that ambiguity.

The platform team's customer list grew faster than its mandate

Business users and data teams are platform customers in roughly half of organizations, at 48% and 49%, and agents in 29%. 43% of respondents expect the role to move toward governance and guardrails, and 9% expect it to be automated away.

Executive summary

The hardest thing to do in platform engineering, and in defining reports on platform engineering, is to decide where to start. For the first time, however, it is simple. We start with the two numbers that tell us everything. 38% of organizations now ship at least twice as much as they did before AI, but only 8% can point to a return that is worth writing home about. Everything in this report sits in the gap between those two figures.



Start with what organizations actually do rather than where they place themselves. Agents now write code and open pull requests for human review in nearly a third of organizations, and a further 18% have them running multi-step work or operating with real independence. However, the largest single group, 41%, are those who still use AI simply to type faster while humans execute everything. Twelve months ago, more than 95% of the industry was in that last group.

Roughly half the industry has matured massively since last year. To understand the maturity of organizations, this report uses four levels of agentic development, from Level 1 (AI suggests, a human executes) through Level 4 (agents initiate and complete work inside human-designed guardrails), based purely on how much work an agent is trusted to carry. The levels predict outcomes better than industry, headcount, or which model an organization has bought, which is why the rest of this report is measured against them.

Maturity for its own sake is not the goal, and the throughput numbers on their own are not encouraging. 39% of organizations report only marginal gains in throughput with less than 20%, and 23% report no measurable change at all. Broken down by maturity level instead of averaged across the industry, the picture changes completely.

What stands in the way of value for most organizations? We identify four things. Absorption is the first. 35% of organizations still review every AI-generated change by human reading alone, so generation has industrialized while the human capacity to check it has not. ROI is the second. Throughput and money are not the same thing, and a fifth of organizations cannot say what their AI spend returned because most teams are not measuring it. FinOps is the third. Token spend prices the attempt rather than the outcome, which means the exact mechanism that produces the throughput - the validation loop retrying until a gate passes - is also the mechanism that runs the cost meter fastest. Governance is the fourth and the sharpest finding in the whole survey. More than half the industry cannot reliably tell an agent's action from a person's, because a quarter of organizations still run agents on borrowed human credentials and a further 30% share a single service account or cannot say how agent identity is managed at all.

35% of organizations still review every AI-generated change by human reading alone, so generation has industrialized while the human capacity to check it has not.

There are organizations rising through the levels of agentic development, and blasting through these four problems. Many of those organizations are doing this through the Agentic Engineering Platform (AEP). It is an evolution of the Internal Developer Platform (IDP). It is what the IDP becomes when agents are treated as users of it rather than tools bolted onto the side of it. And the organizations that build one, and enable the four things agents need together - a dispatch path, a validation loop, scoped identity, and sandboxed environments - are the few that see throughput and ROI move together.

At the same time, successful organizations are increasingly not confined to serving software teams. The substrate an engineering organization builds for its own agents, identity, execution, evaluation, cost attribution, and observability does not know or care whether the workload in front of it is a pull request or a legal claim. What changes per function is the tooling and the paths built on top, not the infrastructure underneath. Which means that the organizations that treat this as engineering infrastructure rather than an engineering advantage are the ones positioned to carry it into every other function in the business.

What is fundamental in this report is that just as last year we witnessed the greatest adoption of new technology in history, now we are witnessing some of the fastest maturity of its use. The number of agentic enterprises succeeding with AI grows every day, and for those teams still struggling, the playbook for success is becoming increasingly clear. Thus, in this report, we are detailing perhaps the greatest turning point in history, as adoption becomes utilization, and utilization sets the stage for unparalleled change in our organizations and the world.

Introduction: What we said, and what happened

Every few months, a new model arrives that is faster, cheaper, or more capable, and it can do things the last one could not. This has produced two kinds of organizations. The first is waiting. It is waiting for a model good enough to fix the ROI it has not found, the costs it can't explain, and the governance it hasn't built. The second is building. It is crafting what the model needs to deliver value now, assuming new capability will keep arriving, and building the organization and platform around that model that will continue to push them to greater and greater heights.

Where Volume 1 left off

Twelve months ago we found that 89% of platform engineers used AI every day, but overwhelmingly for basic code generation and documentation, and overwhelmingly again through AI assistants and copilots.

This was an extraordinary and rapid adoption, but it was also a superficial victory. The data showed that despite all that usage, it rarely converted into organizational return. We identified that gap as being caused by the AI implementation plateau, the stage where organizations stop seeing rapid gains from AI adoption and face slower ROI, integration hurdles, and the need for deeper cultural or process change. It explained why an industry that was using AI daily, mandating it from the top, and convinced it was on the edge of something, saw so little come back out. The big viral number last year was that 95% of organizations surveyed saw barely any return from their use of AI.

However, there was another 95% number last year. One that wasn't fully articulated, however, as the framing for understanding it didn't yet exist. 95% of organizations were at Level 1 or below of agentic development.

LEVEL 0 **Human is the loop**

No agents in the value stream. The internal developer platform baseline, and the foundation everything else stands on.

LEVEL 1 **Human in the loop**

AI suggests, humans execute. Assisted coding, not yet agentic development.

This framing for understanding organizations' maturity in agentic development only fully emerged in April, 2026 in The four levels of agentic software development in the enterprise, but it articulated exactly what we were grappling with the last edition of this report. We will dive far deeper into the other levels in later chapters of this report.

The data was clear last year that those organizations breaking through the AI implementation plateau held platform engineering in common as their differentiator. Platform as a product, golden paths, and standardization allowed them to do far more than their peers with AI. But the details of how platform engineering enabled their success were still solidifying. What has become clear over the last twelve months is how platforms enable that success. That is what this report hopes to demonstrate. It will break down exactly how platform engineering ensures agentic development delivers on its potential, what the leading organizations are doing, and what the state of the industry is for those working to not be left behind.

What the winners were doing, and why it is now the baseline

What those leaders were doing looks obvious in hindsight and, in many cases, looked eccentric at the time. They began to treat the Internal Developer Platform (IDP), the foundation of platform engineering, as the deterministic frame that agents operate inside, rather than as legacy to be replaced, and then extended it across the lifecycle. Those leading organizations were building into their IDP things like agentic paths, agent harnesses, evaluation suites, and governed access to models.

Terms that last year felt like fiction to many readers, but now are common parlance. With that rapid maturity, that set is no longer an advantage. It is the entry requirement. The survey demonstrates that explicitly. Platform teams are now tasked with central LLM connectivity in 45% of organizations, retrieval or chat interfaces in 35%, agent runtimes in 27%, and inference workloads in 27%. Another 15% expect that work inside six months, while only 15% report no AI mandate at all.

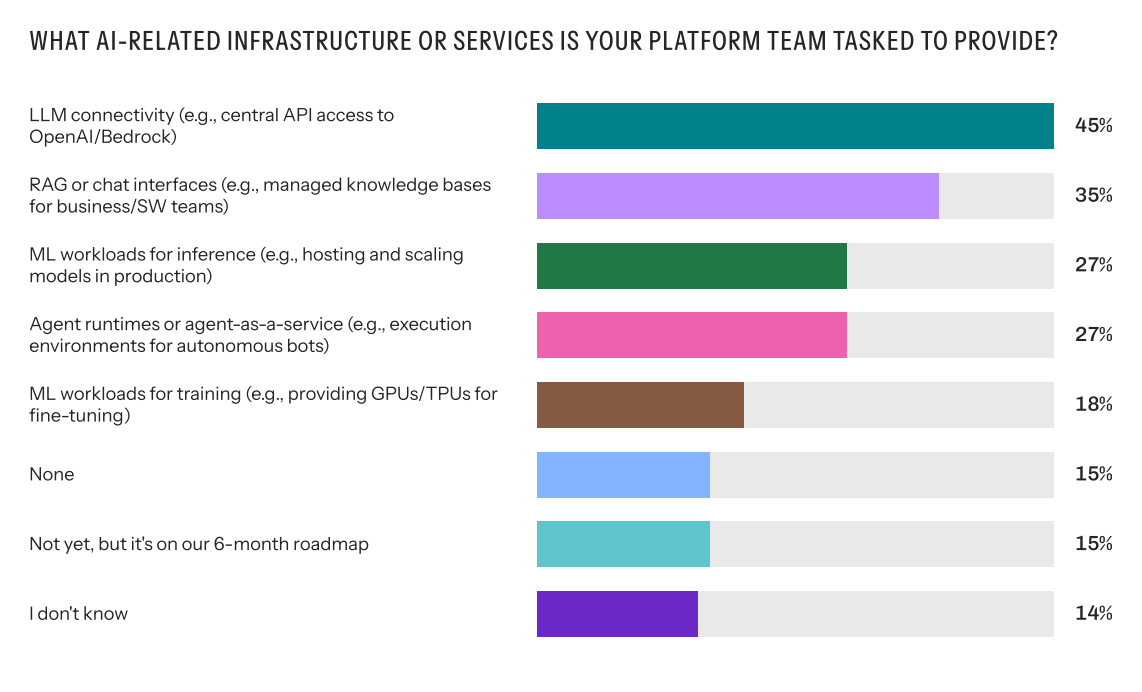


Figure 1: AI-related infrastructure or services provided by the platform team. Multiple answers permitted, so shares do not sum to 100%.

The lifecycle changed shape

At the leading organizations, agents have definitively moved into the core stages of the development lifecycle, writing code, reviewing it, and testing it, while humans stayed on the gates between those stages. The stages do not disappear, and the sequence of them doesn't change - merely the actors do.

This is a different proposition from merely adopting tools, and organizations have committed accordingly. Companies everywhere are doing their best to push this change. 36% now run AI as a company-wide productivity initiative and 33% as a C-level strategic one, while grassroots adoption by individual engineers has collapsed to 6% and only 3% report no activity at all. One of the defining security problems of last year, engineers emailing themselves code out of unapproved AI usage, has largely evaporated. AI is far less often smuggled in through the side door.

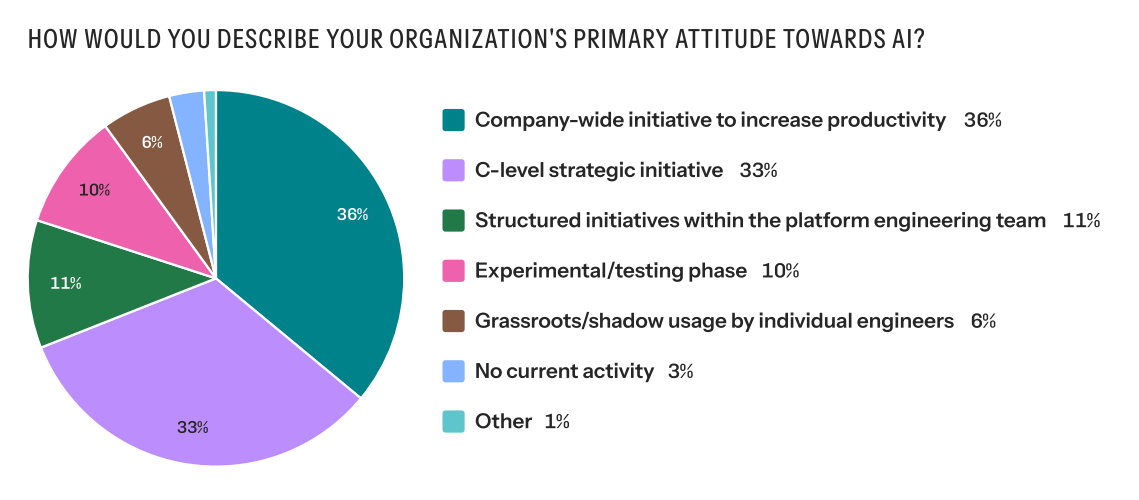


Figure 2: Organizations' primary attitude toward AI.

And while the previous report described engineers experimenting ahead of their organizations, we are seeing now the exact inversion. Organizations that have committed publicly and expensively, ahead of their platforms.

The core use cases haven't changed. Code generation still leads at 84% and documentation at 79%, both up only modestly from last year. The real movement is actually underneath them. Error and log analysis jumped from the low forties to 66%, and infrastructure-as-code generation to 65%. While writing and responding to messages held flat in share and fell from third place to fifth. The easy uses are being displaced by harder ones.

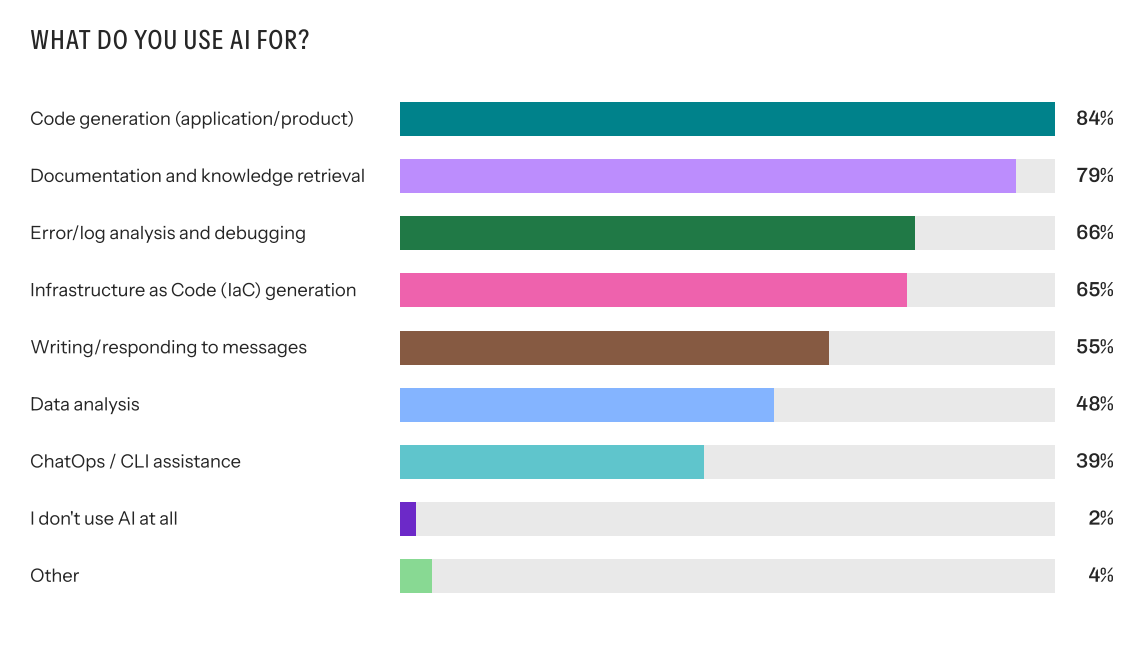


Figure 3: Primary AI use cases across the SDLC. Multiple answers permitted, so shares do not sum to 100%.

At the same time, how organizations consume LLMs has not kept up with what they say they fear (64% mentioned data exposure through public AI APIs and losing control of it as a major threat). 52% go straight to a model provider API and 26% use aggregated hosting, against 11% self-hosting open-source models and 3% using a sovereign provider. The gap, however, is not hypocrisy or inconsistent reporting. It is what happens when teams understand a risk, but lack the surrounding governance to do anything about it and so must (for now) accept it.

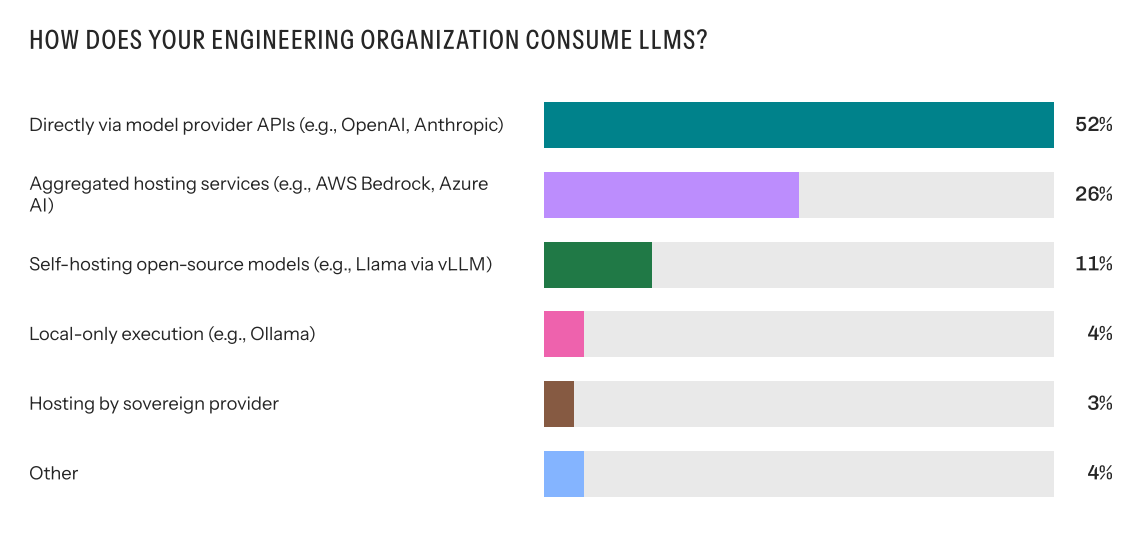


Figure 4: How engineering organizations consume LLMs

Meanwhile, when every organization can reach a frontier model through an API, the model stops being the thing that separates outcomes. It ranks fourth on this survey's own list of obstacles, which is the strongest evidence that the differentiator sits somewhere else (see Figure 11). It's not about what level of access we have to better models; it's about the platforms organizations build around them.

The continuity underneath

None of this is a new class of problem for platform engineering. The discipline exists because recurring work in software development, when done by humans, was being relentlessly ad hoc, at different standards, by everyone separately, and the answer then was to industrialize it. Define the path once, govern it, and let everyone run it. That method is what is now being pointed at agents. And the teams that spotted that, and are doing it already, are the teams racing ahead of everyone else both last year and still to this day. The industry didn't need a new idea on how to manage AI-driven development. It needed to apply the one it had already proved, to a new user that never gets tired, never asks a colleague, and, well, never uses judgment.

Where organizations actually are

It’s important to understand that agentic development is not a destination for teams to achieve. That framing doesn’t make sense, and produces bad behavior within teams. It makes it seem like agentic development is an end goal you can buy or achieve in a few quick steps. No organization becomes agentic on a Tuesday after a good quarter of OKRs. It’s fundamentally a journey, not a designation, and it gets further along, path by path and function by function. The useful question is which step you are standing on and what specifically is stopping you from climbing the next one.

THE FOUR LEVELS OF AGENTIC SOFTWARE DEVELOPMENT

The levels predict outcomes better than industry, headcount or model choice do, because they measure one thing. How much of the work an agent is trusted to carry, and where the human sits relative to the loop. Nothing about the model, the vendor or the budget.

LEVEL 0 — **Human is the loop**

No agents in the SDLC. The internal developer platform baseline, and the foundation everything else stands on.

LEVEL 1 — **Human in the loop**

AI suggests, humans execute. Assisted coding, not yet agentic development.

LEVEL 2 — **Human on the loop**

Agents generate pull requests in parallel. Humans trigger the work and verify behavior rather than inspect every diff. The first real parallelization.

LEVEL 3 — **Humans as orchestrators**

Agents work continuously from observed signals, and review becomes exception-based.

LEVEL 4 — **Fully autonomous**

Systems of agents initiate and complete work inside human-designed guardrails.

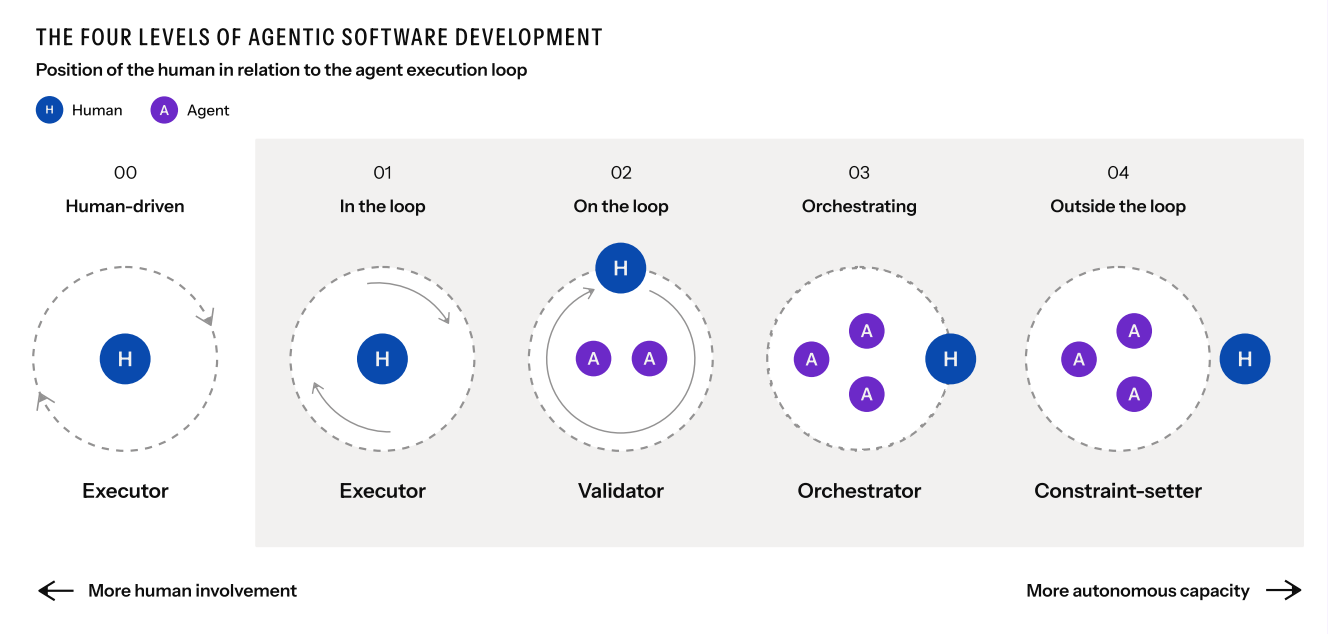


Figure 5: The four levels of agentic software development

An important caveat for when we assess where teams are at on these levels. The levels are self-reported, andnself-reports often run optimistic, so the ground truth is very likely worse than what follows.

The maturity snapshot

Every organization has an intuition about how far along it is, but ask an engineering leader how advanced their AI usage is, and in many cases the answer comes back as a list of tools. This only tells you what the company bought but far less about what it does with them. Some read their pilot programs as proof of maturity. Others read their production incidents as proof of nothing at all. The four levels exist to help settle that argument with clear numbers that rely more on usage rates, than tool lists.

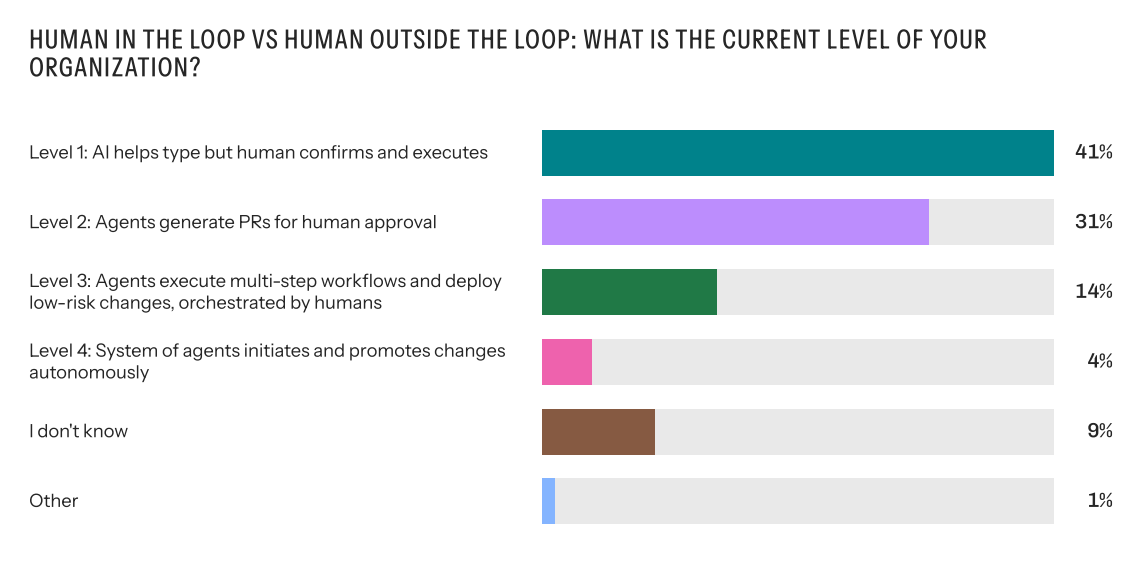


Figure 6: Self-reported agentic maturity, by level

Level 1 holds the largest single group in this survey at 41%. Level 2 follows at 31%, Level 3 at 14%, and Level 4 at 4%. A further 9% can't place themselves at all. That is not, however, due to issues with the data; it is because agentic maturity is not yet a property one organization in ten measures.

When we take that data against the previous report, that is a staggering amount of movement. Twelve months ago, more than 95% of organizations sat at Level 1 or below. Today only 41% do. Very little in this industry moves that far that fast, and anyone still calling agentic development a fad should sit with that chart for a moment.

Then after we're done clapping ourselves on the back, look at where the movement stopped. 31% reach Level 2 and are parked, while only 18% operate above it. Level 2 is the first level that demands platform work rather than better tooling, which is exactly why the industry has bunched up underneath it.

At the same time, there is another point holding us back from celebration. Thirteen of the fourteen organizations reporting Level 4 have fewer than 100 engineers. And looking at it from the other end, among organizations with more than 100 engineers, only a single one reports Level 4.

That means our argument has a significant problem that needs shouting out. If agent autonomy today is mostly a small-team achievement, one reading is that the platform is a scale tax rather than an enabler. The more likely reading, in our opinion, is that just as it was in the early days of platform engineering, a small team can standardize its work and assign clear ownership in an afternoon, while a large enterprise is doing the same job against legacy estates, competing standards, and change-management problems that no architecture can remove. The important data point from Volume 1 last year to today was "did more organizations break through the AI implementation plateau". The important data point going into next year is "do more organizations with over 100 engineers move higher in the levels of agentic development".

The turning point

Volume 1 reported that 95% of organizations were seeing nearly zero return from AI, which was connected to the plateau. If more organizations breaking through is the key change to look out for this year, let's see if the data shows it.

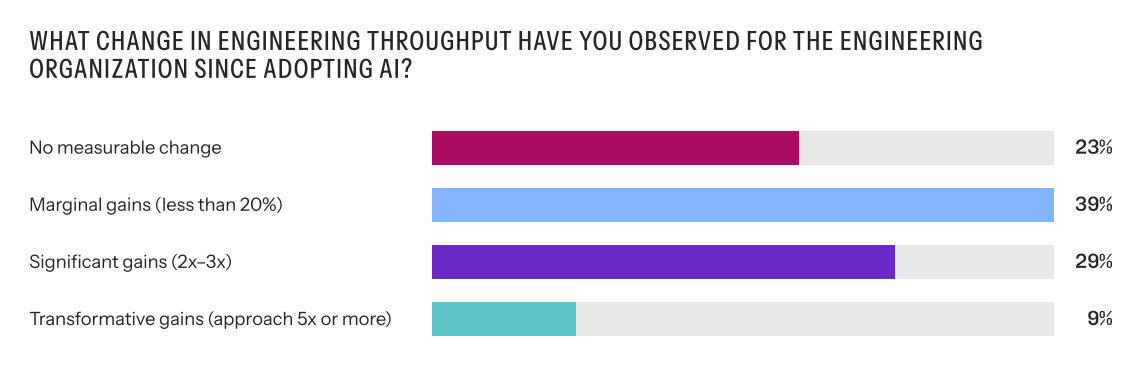


Figure 7: Change in engineering throughput since adopting AI

Starting with throughput, the reported improvement spans the entire range. 39% see marginal gains under 20%, 23% report no measurable change at all, while 29% report gains of 2x to 3x, and 9% report gains approaching 5x or more. The same technology, available to everyone, is producing wildly different results.

It's clear that something other than the technology is deciding the outcome. Not the model, not the budget, not the industry. It's what each organization built underneath and around its agents that is the differentiator.

Level	Base (n)	Reporting at least doubled throughput
Level 1	143	30%
Level 2	110	36%
Level 3	50	60%
Level 4	14	79%

Figure 8: Share reporting at least doubled throughput (2x to 3x, or 5x and above), by agentic maturity level.

At Level 1, 30% report at least doubled throughput. At Level 2, 36%. At Level 3, 60%, and at Level 4, 79%.

Notice the change between Level 1, where AI assists but no agents are in the value stream, and Level 2, the first agents. Putting agents into the value stream buys only a measly six percentage points. But move on to level 3? Well, building the platform they run on buys twenty-four.

Level 3 is both where we see a spike in throughput and a collapse in those at that level. Why is that? Well, Level 3 requires governed, machine-executable paths across code, review, and test at the same time, because agents working continuously from observed signals will touch all three.

Those capabilities also don't arrive one at a time and pay out incrementally. They're something that arrives together and thus pays out together, which also means a half-built set carries most of the cost and almost none of the actual benefit. Your agents might be excellent at generating code, but if review and test are still manual, you are getting very little of the success. Meaning that though the lifecycle might have changed shape, it's only the organizations that rebuilt the platform to match the new shape that get the benefit.

There are two honest caveats to this analysis. Both the level and the gain are self-reported, and it is possible that organizations already converting AI into throughput are the ones with the resources to reach Level 3.

What the throughput bought

There is one brutal caveat under all of this. Throughput is not return. It may be the raw material of return, but it alone does not guarantee or demonstrate ROI.

The gap is not that the output is not real. It is that shipping more features is not the same as delivering more business outcomes, and most organizations are still measuring the first as if it were the second.

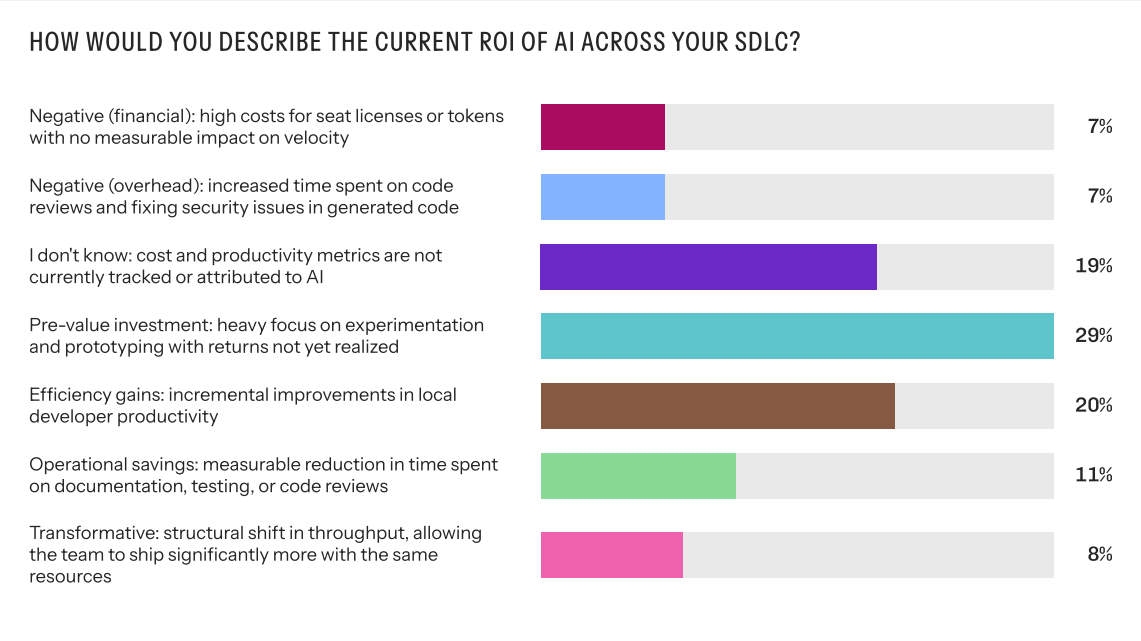


Figure 9: Current ROI of AI across the SDLC

Only **8%** of organizations describe the impact of AI across their lifecycle as transformative.

29% are still pre-value, deep in experimentation with nothing realized yet. The rest are spread across local efficiency gains, operational savings, and 14% that are even negative once licenses, tokens, and review overhead are counted. Perhaps worse than negative measured ROI is that a fifth of organizations cannot answer the question at all, because cost and productivity are not tracked or attributed to AI.

Broken out by level, the return follows the same shape as the throughput. Only 2% of organizations at Level 1, where agents aren't yet utilized but copilots & AI assistants are, describe the impact of AI as transformative. While 9% at Level 2, and more than 20% at Level 3 or Level 4. That means that more than 1 in 5 organizations at Level 3 or above report their ROI as transformational.

Something specific again unlocks at Level 3. It's the point where the platform can run work continuously, while still accounting for what comes back, which means the increased output is able to be successfully absorbed.

Bleak as that reads, it is enormous progress. It's also the strongest evidence in this report that the platform work pays. Volume 1 described an industry where adoption was near universal and return was near zero, with the plateau sitting squarely between using AI and getting real value back. That wall is coming down. 38% of organizations now ship at least twice as much, and where Volume 1 found return close to zero, a real share of the field now reports efficiency gains, operational savings, or a structural shift in what it delivers. While 8% report incredible returns.

More importantly, this also highlights that the path to those returns is more clear than ever.

Platform readiness, not model capability

The topic that garners the most attention by far in the discourse around AI is the capability and advancement of models. However, when asked what most obstructs scaling AI, only 1 in 10 of those surveyed felt it was model capability that was holding them back. While in actuality it was platform readiness that topped the list at 27%. While token and infrastructure costs came second at 23%, and human review bandwidth third at 16%. The differences between the majority of organizations and those agentic enterprises at the top is not who has the greatest access to models, but those who have the most mature platforms that can enable those models in delivering transformational ROI.

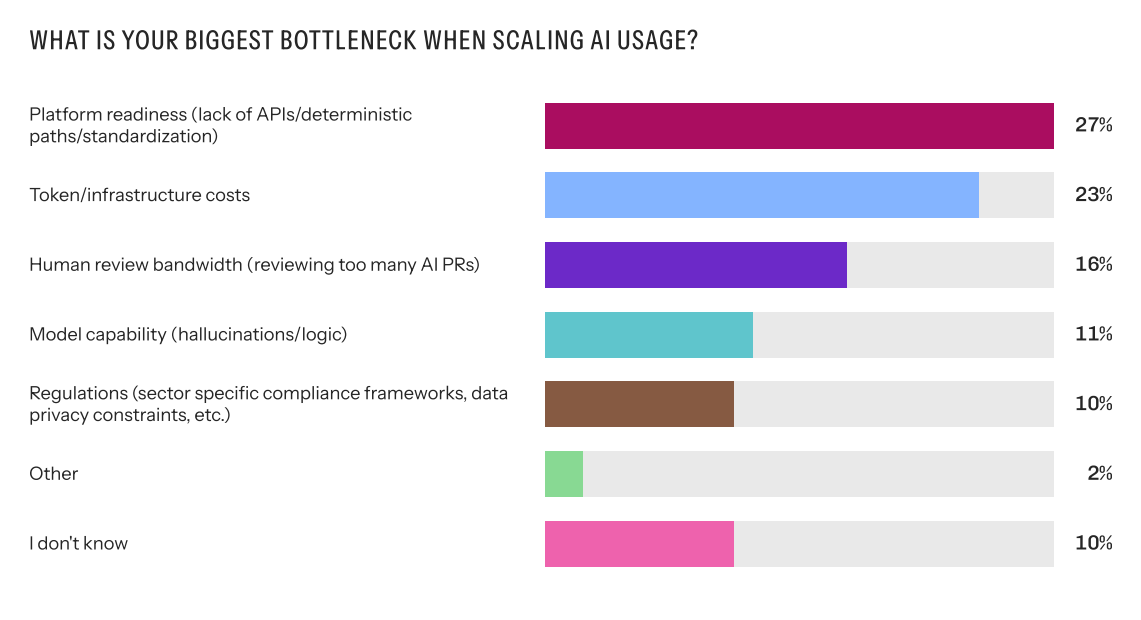


Figure 10: Biggest bottleneck to scaling AI usage

> Every week, a new, better model comes out. And we're all using the same models. Why are some organizations 10x faster than other ones? While at the same time being more secure with fewer incidents. It's because they have the platform around the model. The organizations that can't keep up or constantly face AI security and governance problems, are the ones whose platforms have been outpaced by the model.
>
> Rickey Zachary
> GLOBAL ENGINEERING PLATFORMS LEAD, THOUGHTWORKS

Quality, and the weak point

One of the other core ideas in last year's report was the fear or challenge of low-quality AI-generated code. This was reported both in the survey data and extensively across dozens of expert interviews. Despite that, the quality panic that dominated early skepticism of AI appears to not be materializing.

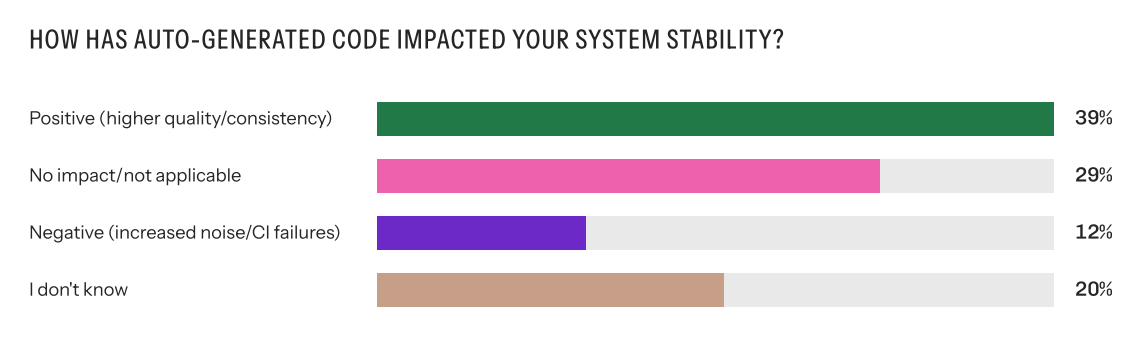


Figure 11: Impact of auto-generated code on system stability

Surprisingly, 39% of organizations report that AI-generated code improved stability and consistency, against 12% seeing more noise and CI failures. 29% report no impact, and 20% do not know what effect it had (which is its own finding, for many orgs output is arriving faster than its consequences are being measured).

The real weakness, however, is downstream of quality. Bad output is cheap to reject. Output that is mostly fine has to be read properly, every time, and there is a great deal more of it.

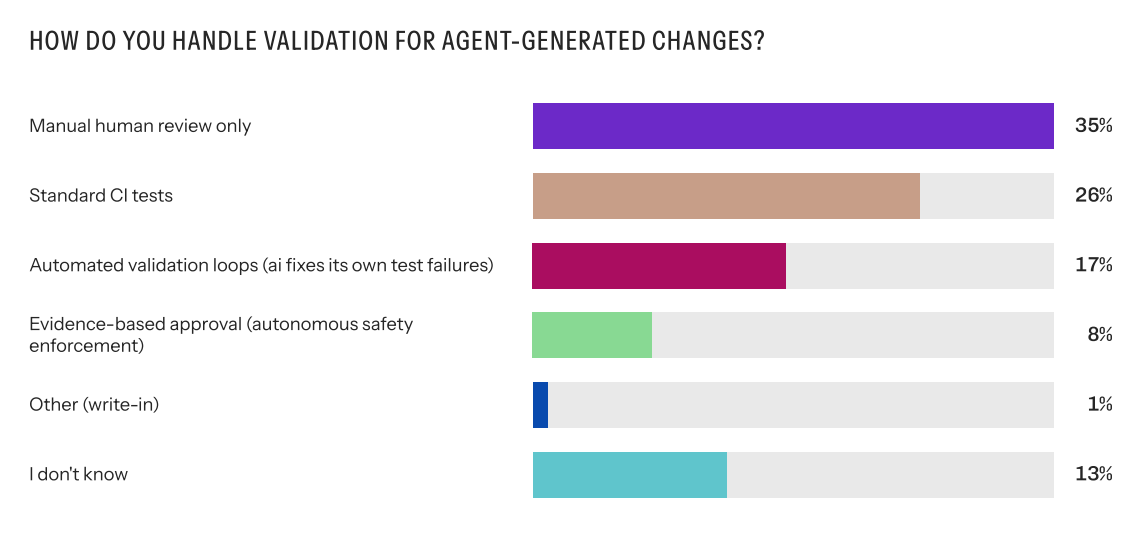


Figure 12: How organizations validate agent-generated changes

This is hugely problematic as 35% of organizations still check AI-generated changes by human reading alone, while only 17% run automated validation loops. A further 13% do not know how those changes are validated at all. This means that for many organizations, whether they are producing code via AI-enhanced developers or through agents, the generation might have industrialized but absorption has not, and that load is falling massively onto human heads.

AI has made code cheap and judgment expensive. Research confirms that 97% of developers either already coding with AI or has tried an AI coding tool. Yet in reality, far fewer organizations can govern, secure and run what comes out of it. The bottleneck moved from writing software to shipping it responsibly; which puts developers at the center of value creation. The scarce skill was never typing the function but deciding what to build, what to trust, and what to own in production. That is why the platform choice and eco-systems including databases and security measures belongs with them. Demos are easy; the work after the demo is ensuring the eco-systems are ready. And that ecosystem needs flexibility across models, modalities, and workflows, with no lock-in and with cost efficiency (incl. tokenomics) designed in from the start. That's where enterprise software succeeds or fails, and that is the ground we've chosen; trusted cloud, developer platforms, security, governance and business context in one layer that carries a developer's judgment all the way to production.

Irina Gontcharova

CLOUD AND AI LEAD EMEA, GM - MICROSOFT

The Agentic Engineering Platform (AEP)

The biggest development of the last year is that the thing to build now has a name. The Agentic Engineering Platform (AEP). It is a platform on which humans and agents develop software together.

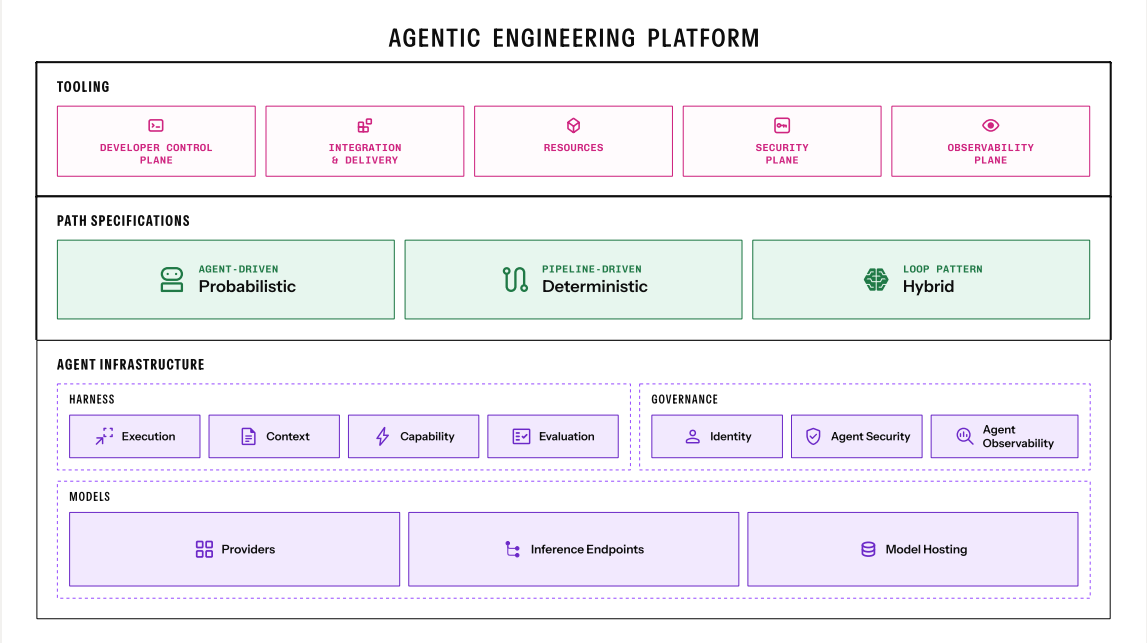


Figure 13: The three layers of an AEP, with the tooling layer marked as the IDP and the agent infrastructure layer marked as the substrate that carries agents anywhere in the enterprise.

Agents function similarly to human users. They require identity, pathways, policies, environments, and destinations for outputs. For over a decade, platform teams have provided all of these elements to human developers. The challenge arises when a developer encounters an ambiguous policy and seeks clarification from someone. In contrast, an agent will simply proceed on its own. If a developer notices an issue, they can halt their actions; however, an agent lacks this instinct and operates in parallel with multiple other agents (sometimes nine, sometimes 90). Consequently, an agent without specially designed controls and validation checkpoints does not fail gracefully; instead, it generates chaos at machine speed. This is where the AEP comes into play.

> The Agentic Engineering Platform is the evolution of the IDP, not the replacement of it. It builds on the foundations of platform engineering that we have all been building and enabling over the last few years.
>
> Kaspar von Grünberg
> AUTHOR, THINKING IN PLATFORMS

Three layers

The tooling layer on top is the five platform planes teams already run. Developer control, integration and delivery, resource, security and observability, now under agent-scale demand. What changes here is load rather than structure, and it turns reliable APIs, predictable rate limits, structured outputs, and explicit failure modes from good practice into preconditions.

The middle layer holds the paths themselves. A path is a repeatable route from intent to usable output. What the AEP adds is that those paths become governed and machine-executable, and invokable by a human or an agent. The agent infrastructure layer underneath carries identity, context, capability, execution, evaluation, security, and observability, all defined as code. Deterministic paths run on the tooling layer alone, with no agent infrastructure beneath them.

SIDEBAR

Path types, and what each one costs to govern

Deterministic

Pipeline-driven, same output for the same input. Builds, scans, policy gates, promotions, deployments. Governance is cheap because the behavior is knowable in advance, which makes this the correct default for anything auditable, irreversible or regulated.

Probabilistic

Agent-driven, returns a plausible output. An agent reviewing a change, drafting a specification from a ticket, triaging an alert. Correctness cannot be established by inspecting the path definition, so the governance cost moves into the evaluation apparatus. Evals become the test suite!

Hybrid

Probabilistic generation wrapped in deterministic gates, looping until the gate passes. Higher token cost per unit of work, full auditability at the gate, and blast radius bounded by whatever the gate refuses to let through.

Choosing between the three is the crucial design decision and leaving it implicit is how organizations end up running ungoverned probabilistic paths in production. We recommend naming the type of every path the platform exposes, and requiring deterministic gates on anything with an irreversible effect.

The Internal Developer Platform (IDP) is the deterministic frame, not the legacy layer

The AEP adds layers to the Internal Developer Platform (IDP). It does not replace it or route around it. An organization with a mature IDP is not starting again; it is continuing, and an organization without one has nothing to build on.

And, not every organization needs the whole thing on day one. At Level 1, the agent infrastructure can be thin, because humans still catch everything and most paths stay deterministic. At Level 2, agents produce in parallel and validation has to become a loop rather than a gate, which makes identity, workspace provisioning, evaluation, and agent security necessary at the same time. Building that set is the work that carries an organization out of Level 2, and by Level 3 the full substrate has to be running.

Organizations that throw agents at a Level 1 substrate will find their agents moving faster than their guardrails. Those are the failures that reach the front page. They are also frequently reported as model failures, but they are, at their core, platform failures.

The validation loop is the engine

A validation gate is a checkpoint a human walks through. Continuous integration runs the tests, a reviewer inspects the change, and approval is binary. That will be a familiar model to everyone reading this report.

A validation loop is the same model but as an industrial feedback mechanism. When validation fails, the failure signal routes back to the agent, which revises the implementation and retries. The cycle repeats until the deterministic checks pass. By the time a change reaches a human, it should almost always pass, and where it does not, the failure becomes feedback that improves the agent's operating model. This means that first-pass failure is convergence, not defect.

This is the answer to the deluge in front of the human we highlighted above. Without the loop, every additional agent joins the same queue, and human review bandwidth caps throughput no matter how many agents run. With it, the cap lifts, review moves to exceptions, and output starts converting into delivery rather than into backlog. That single change is most of the distance between shipping twice as much and being worth twice as much.

One task, end to end.

In an organization that has made this work, an engineer picks a ticket and dispatches it rather than opening an editor. The platform packages the context the agent needs, binds an identity that belongs to the agent and not to the engineer, provisions a disposable workspace, and lets it run. The agent produces a change. Deterministic checks evaluate it, fail it, and hand the reason back. The agent revises and tries again. By the time a person looks at it, the change has already cleared everything a person would have checked, and what the engineer reviews is the evidence rather than the diff. Fundamentally, nothing in that sequence is a model capability. All of it is platform.

SIDEBAR

The ASDLC, and how the AEP enables it

Just as the AEP is an evolution of the IDP, the Agentic Software Development Lifecycle (ASDLC) is an evolution of the SDLC. It's the same lifecycle with agents now doing the coding, reviewing, and testing, and humans holding the gates between those stages. Those gates aren't a holdover from the old process. They're what makes it safe to run probabilistic work.

The ASDLC describes the flow, which stages exist and in what order work moves through them. The AEP is what makes that flow executable.

Some stages stay exactly as pipeline-driven as before: CI builds, security scans, deployments, policy gates. Other stages are where the agent now does the work: drafting specs from tickets, reviewing pull requests, generating release notes. Underneath the agent infrastructure layer supplies what a path needs to work: a harness (execution, context, capability, evaluation), governance (identity, security, observability) and the models. The lifecycle tells you what has to happen. The platform is what makes it happen.

Neither replaces platform engineering; both are practices within it. Harness engineering lives inside the harness, tuning the machinery of a single agent turn from execution through evaluation. An agent, in this framing, is a model plus the harness. Loop engineering lives one layer up, tuning what happens between turns: the gate verdict, the feedback that goes back, the stop condition. The harness runs one agent. The platform governs the fleet: identity, security, and observability, owned once and enforced everywhere.

What is in the way: Absorption, ROI, FinOps and governance

Knowing what to build is not the same as being able to build it, and the organizations climbing this ladder hit the same four things in the same order. More output than they can take in. Output that does not convert into money. Costs that most organizations aren't measuring at all. And, underneath all three, an inability to see or attribute what their agents are doing. All four are foundational platform problems.

Absorption: More output than review can take in

AI speed is officially crashing against everyone, everywhere, all at once. Code, reviews, logs, vulnerabilities, specifications, documentation.

This is happening within our own organizations and across the entire industry. Open-source maintainers are absorbing rising volumes of machine-written contributions. Specifications run longer than anyone will read. Documentation is generated faster than it can be curated or trusted. Security findings increase with every line of generated code. Observability tells the same story in telemetry, where each model call throws off several times the signal of a conventional log event against budgets that did not move. Every artifact an engineering organization produces is arriving faster than the people and systems downstream can consume it. Same shape every time. Enormous new supply, no capacity to receive it.

On the plus side, generation is solved. Congratulations. That is also the entire problem.

With the single largest usage of AI being code generation, it's no surprise that it is where this problem hits hardest. Writing code was never the only bottleneck, and the theory of constraints has held for decades that optimizing anything other than the bottleneck produces nothing. Generation got faster, so the work moved rather than disappeared, from typing into reviewing, and the larger pile landed on the same people. Recall how organizations validate agent-generated changes highlighted above, a third of organizations read every AI-generated change by hand, and only a quarter have anything autonomous in that path at all.

When we compare those numbers against teams that report as level 1 or level 2, the gap becomes even more clear. Review bandwidth as the single biggest bottleneck doubles the moment agents enter the value stream, from 12% of organizations at Level 1 to 23% at Level 2. As soon as agents start working in parallel, teams smash into this review wall.

Set all this against 38% reporting at least doubled output with only 8% calling the return transformative, and there is no mystery why that throughput to ROI gap exists.

The people who were the quality gate become the designers of the quality gate, which is a different job requiring different skills, and most organizations have not yet said so out loud to the people concerned.

ROI: Why the throughput did not convert

Absorption explains why extra output does not get delivered. It does not explain why delivered output does not show up as money.

That is not a reason to discount the throughput. It is, however, a reason to stop measuring value in units of output. This report frames return through six vectors rather than through ticket counts, because they are where the value of platform work has always shown up:

▪ **Delivery velocity**	Meaning how quickly work moves from intent to production.
▪ **Outcome predictability**	Meaning how reliably a commitment turns into a shipped thing.
▪ **Operational resiliency**	Meaning what happens when something breaks and how much of the response is automatic.
▪ **Maintenance overhead**	meaning what proportion of capacity is consumed keeping the existing estate alive.
▪ **Organizational learning**	meaning whether a solved problem stays solved for everyone.
▪ **Structural incentives**	Meaning whether the easy path and the correct path are the same.

The platform moves every one of those, and a throughput number shows only the first. An organization measuring its return in output alone is reading one dial of six. So a company reporting no return may have had none, or may simply never have looked at the other five.

FinOps and tokenomics: The cost most organizations are not measuring

Just over a quarter of respondents could not tell us what their AI usage cost. 13% report that token spend is not tracked at all, and another 14% do not know whether anyone tracks it.

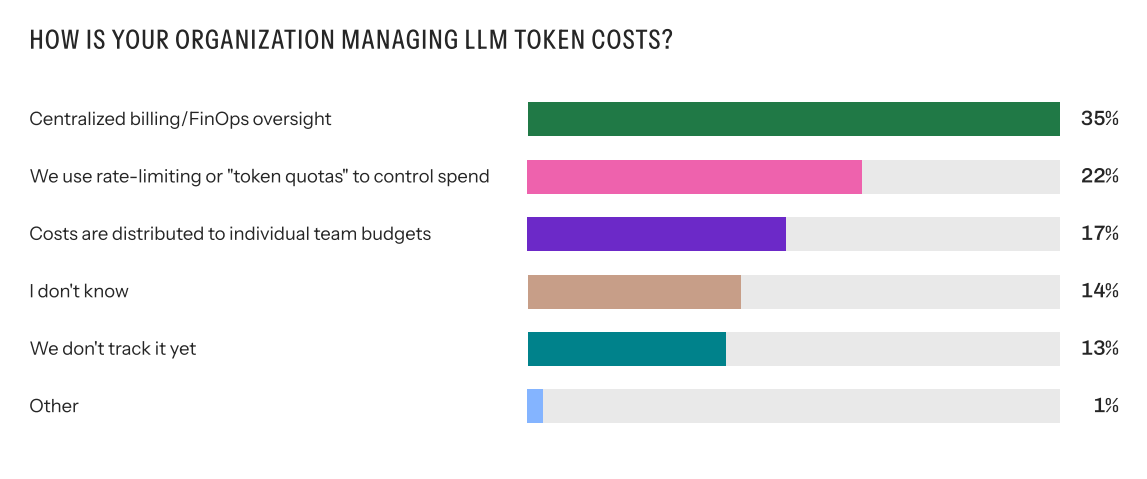


Figure 14: How organizations manage LLM token costs

For those that do track, 35% have centralized billing or FinOps oversight, 22% enforce rate limits or quotas, and 17% distribute costs to individual teams. Those are three different capabilities. Seeing the bill, being able to cap it, and knowing whose it is are not the same thing.

It is thus no surprise that cost was already the second-largest obstacle to scaling AI in this survey, behind only platform readiness, when 13% are not tracking it at all and another 14% cannot say whether anyone is, and the rest are mostly scrambling to control the deluge of token spend.

A big part of the reason teams are struggling to control this new cost domain is that it is priced differently than what they are used to. Software has always been priced by capacity. Servers you bought, seats you licensed, engineers you hired; all of it paid for the ability to do work whether the work happened or not. Token spend, however, prices the attempt to do something.

Now put that next to the validation loop. The mechanism that converts agent output into delivered work generates attempts deliberately until a gate passes.

The thing that buys the throughput is also the thing that massively runs up the meter. Every loop costs tokens. That is not an argument against the loop; it is the reason cost governance is a Level 3 precondition. Without it, the cost impact of Level 3 activity would likely negate any of the benefit. Because work is generated from observed signals rather than human initiation, spend decouples from headcount and can go parabolic.

We see this clearly as 15% of organizations at Level 1 and Level 2 do not track token spend at all. At Level 3, teams running agents continuously tend to always sort out the metering first.

Governing it is also a core platform component, not a policy. In AEP terms it belongs to audit and attribution, sitting in the observability layer and owned by the platform directly rather than by finance. Four things it has to do:

- Attribute every model call to a team, a path, and a model, so all spend has an owner.
- Watch spend velocity rather than spend totals, because the rate of change is the early warning.
- Enforce hard limits in the platform, so a runaway loop hits a wall instead of a monthly invoice.
- Produce its own audit trail, which is the same evidence a regulator will ask for.

None of that is exotic. It is the FinOps discipline platform teams already have, pointed at the first resource that decides for itself how much of itself to consume.

> Every AI-powered workflow now carries an operating cost, and every agent inherits whatever is wrong with the data underneath it. The organizations doing this well start with the data rather than the model, and they treat governance as an architectural property rather than a policy document. The challenge is no longer how to build agents. It is how to govern the resources they consume.
>
> **Shayan Mohanty**
> CHIEF DATA AND AI OFFICER, THOUGHTWORKS

Which suggests the metric to run this on. Cost per token is an accounting number, and it will fall on its own as models compete. Cost per accepted output is an engineering number; it connects spend directly to the conversion problem in the previous section about ROI, and it is the one a platform team can actually improve.

Governance: Who did what, and can you prove it

Absorption, ROI, and cost each look like a separate problem. What organizations say they cannot answer about their own agents suggests they are not. 9% cannot say what level they operate at. 13% do not know how AI-generated changes are validated. 18% do not know how agent identities are managed. 14% do not know which controls apply to generated code. 14% do not know how token spend is handled. 20% do not know what generated code did to their stability. 8% have no monitoring of unmanaged AI use at all. So, somewhere between one in ten and one in five organizations cannot answer a basic question about how agents operate inside them. And what is worse than an agent producing code at the speed of 10 devs with uncapped costs? An invisible agent doing all that and more.

Why the old controls do not hold

Every security control in the enterprise was designed for software that does what it is told. Agents simply don't. They open pull requests, file tickets, update records, and call other systems; they choose their own tools and their own order to reach a goal that was maybe asked for, perhaps not. Speaking at PlatformCon 2026, Caroline Wong, in her talk, "Security by design for agentic AI: How CISOs stay in control of autonomous systems," put it as the flexibility that makes agents powerful being the same flexibility that makes them hard to govern. Which is why last year, those security teams whose response to AI was simply to say "No" were departments that stopped functioning. A veto cannot be enforced at agent speed. Securing agents has to be built into the platform.

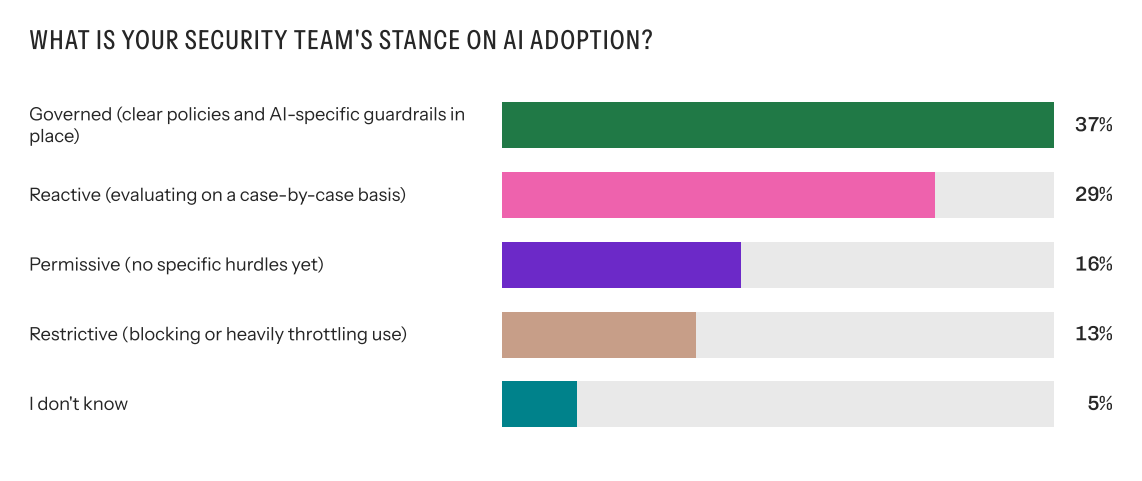


Figure 15: Security team's stance on AI adoption

To the industry's credit, that's exactly what happened. 37% of security teams now govern AI through clear policies and AI-specific controls, and only 13% block or heavily throttle it. One of Volume 1's findings was a gap between stated policy and actual practice. This year

the more interesting movement is that governance has shifted into the platform, where it can run at the speed of the thing it governs. It's worth noting the limit of that progress, however. Volume 1 found 16.9% shutting AI down or making it hard to use, so the blocking contingent has only barely shrunk. It has just been outnumbered rather than converted.

Identity is the weak point

However, governance is only ever as strong as the identity model underneath it, and agent identity is the least reassuring result in this survey. Fewer than one organization in three gives its agents a scoped identity of their own, and only another 19% manage agent identity dynamically through policy. A quarter hand over human credentials instead, and another 12% share a single service account across every AI tool. 18% have no idea.

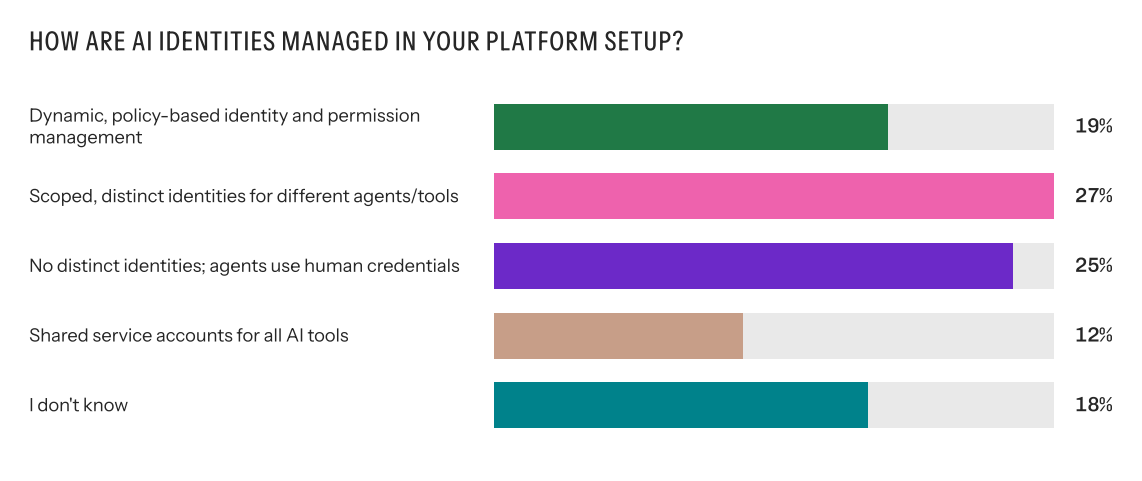


Figure 16: How AI identities are managed in the platform

An agent running on a borrowed human login cannot be audited, scoped, or switched off without switching off the person it borrowed from. That is more than a policy gap; it is an unbounded blast radius that hits almost a quarter of the industry.

You can't govern agents that you don't know exist. And we are already seeing the early version of a shadow agent problem, just like we saw shadow IT and shadow SaaS. Teams are experimenting, developers are building, and new agents are showing up faster than most organizations can track them. So the first question isn't is this agent secure? It's do we even know it's there?

Caroline Wong

SECURITY BY DESIGN FOR AGENTIC AI: HOW CISOS STAY IN CONTROL OF AUTONOMOUS SYSTEMS

The controls in place, and what teams fear

Notice what those controls actually govern. Almost every one of them inspects code: peer review, secret scanning, SAST and DAST, dependency analysis. But 65% of respondents already use AI to generate infrastructure as code, so a growing share of what agents produce does not stay in the repository. It provisions, configures and changes the running estate. With both policy-as-code enforcement and sandboxed execution still in the minority, the controls stop at the boundary of the code path for most organizations while the effects carry on past it.

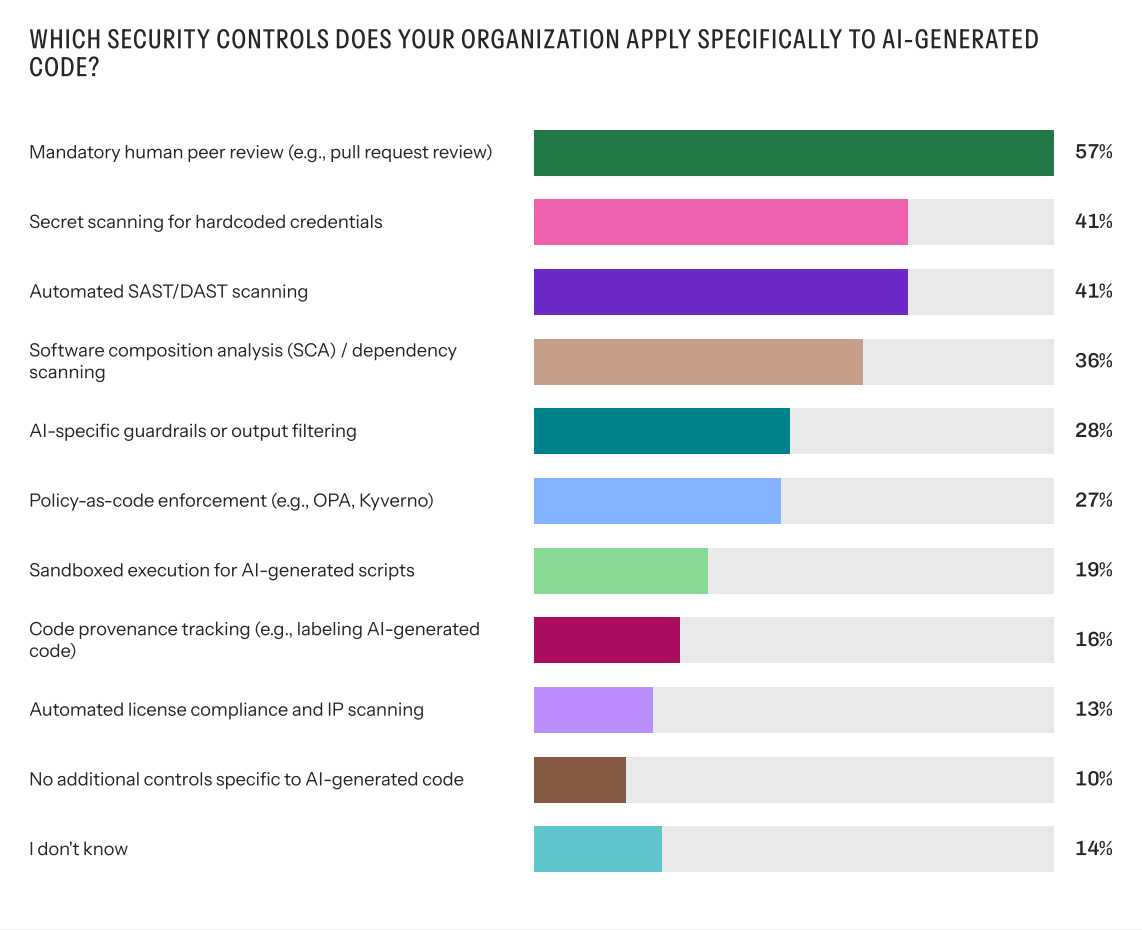


Figure 17: Security controls applied to AI-generated code. Multiple answers permitted, so shares do not sum to 100%.

That is the same gap named by the 43% who cited agents bypassing existing controls as a major threat, and it is why what is declared in code and what actually exists in production come apart faster once agents make the changes. An organization that cannot compare the two cannot tell an agent's change from a person's, or an intended change from an accident.

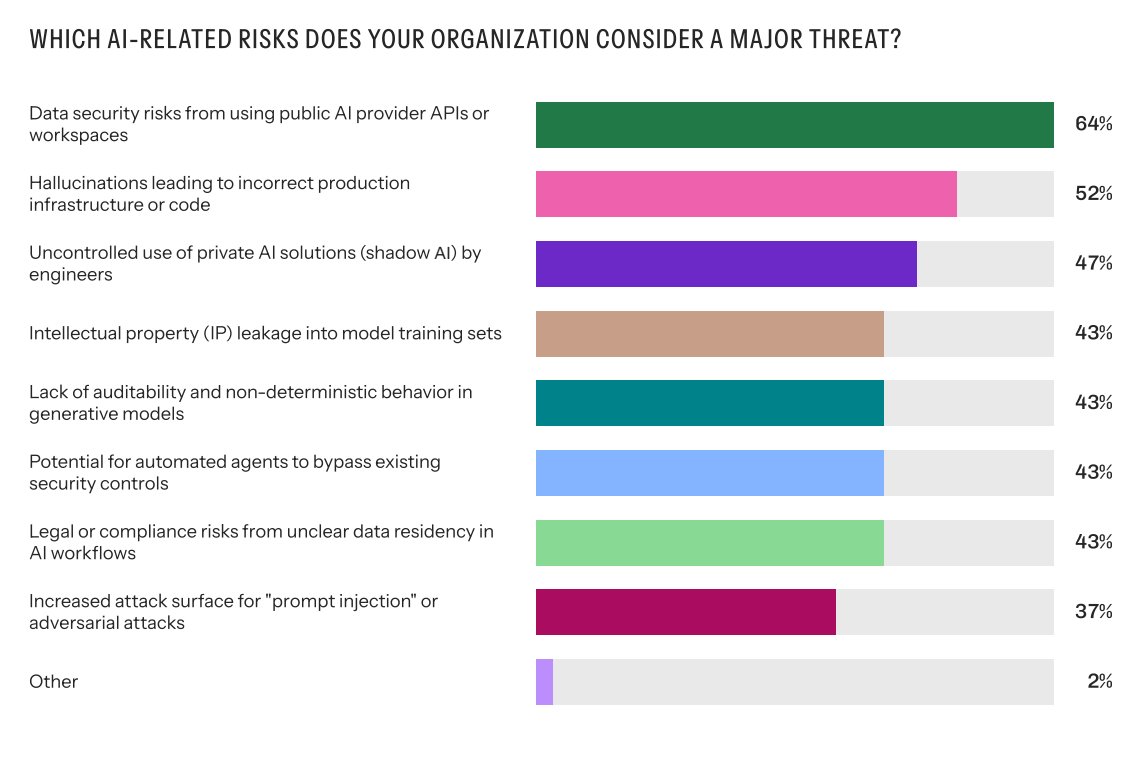


Figure 18: AI-related risks considered a major threat. Multiple answers permitted, so shares do not sum to 100%.

What organizations fear is also massively revealing. Data exposure through public AI provider APIs leads the list at 64%, followed by hallucinations reaching production at 52% and unmanaged tool use at 47%. Then four risks tie at 43%. Unclear data residency and the lack of auditability in non-deterministic systems. Prompt injection ranks lowest of the technical risks at 37%, which is surprisingly low for an attack surface demonstrated repeatedly in the wild.

Shadow AI: you cannot govern what you cannot see

On shadow AI, 47% call it minimal, 27% admit it is significant, and 8% have no visibility whatsoever. This data could be read positively, as Shadow AI was one of the biggest concerns last year, and it appears less widespread than feared.

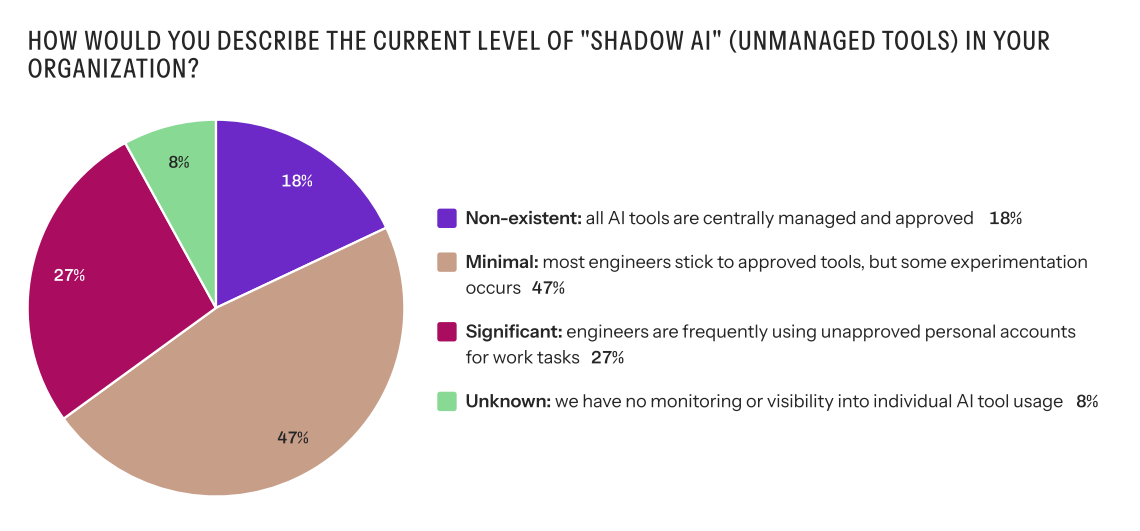


Figure 19: Current level of shadow AI

But you cannot govern what you cannot see, which is why observability has to precede restriction. A control applied to invisible usage governs nothing, and restriction imposed before visibility pushes usage further out of view.

What the leading organizations build

Again, what is working for the leading agentic enterprises is clear. Governance enforced at the workspace level, so every agent process inherits policy, identity and network boundaries from the environment it runs in rather than from whichever framework happened to launch it. Privilege separation between humans and agents, with agents on least privilege and pull-request-only writes. Ephemeral workspaces rather than persistent ones, which prevents context pollution and permission drift and makes parallel execution safe. And controlled failure over silent incidents, so that a blocked action that is logged, attributed, and surfaced is a governance success. This also stops agents from burning tokens retrying something they will never actually be allowed to do.

Every one of those constrains how an agent works, but none of them constrain what it is permitted to achieve. That is why tighter governance actually raises the autonomy ceiling instead of lowering it. Might feel counterintuitive to some of us that stricter governance actually results in higher productivity, but in an agentic future, it's the reality.

The expanding mandate

Everything so far in this report has been about what platform teams build. This chapter is about who they build it for. That list has been growing for five years. Security teams, FinOps, infrastructure, observability, business teams, and of course, AI agents each arriving with their own needs and their own vocabulary.

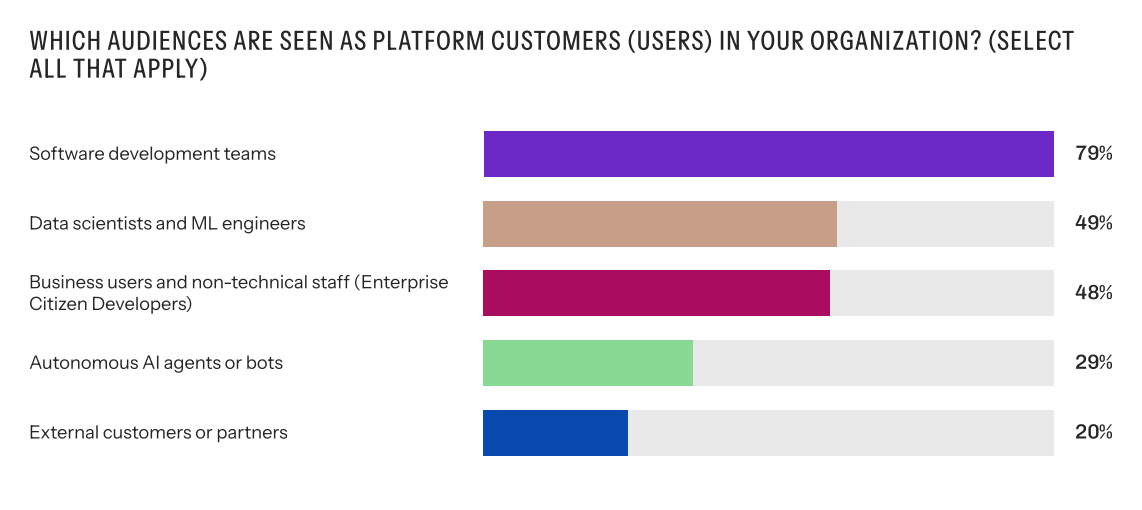


Figure 20: Audiences seen as platform customers Multiple answers permitted, so shares do not sum to 100%.

Software development teams still lead at 79%. Data scientists and machine learning engineers sit at 49%, business users and non-technical staff at 48%, and external customers or partners at 20%. Four audiences with four sets of expectations, served by the platform team that has stretched to cover all of them using the core fundamentals of platform engineering.

At the same time, 29% of platforms now count autonomous agents among their customers. This tracks with where organizations put themselves across the different levels of agentic development. However, it's worth being precise that this is a self-report about who consumes platform services today, not a claim that those organizations operate autonomously as a whole.

QUICK DIVE

Build it, or adopt it

Nearly half of platform teams now own central connectivity to models, which makes one decision unambiguously theirs. Assemble the AI base layer from open components, or adopt a curated stack and inherit somebody else's opinions.

That decision is live rather than settled. The dominant GPU vendors, alongside many of the major AI infrastructure providers, are now pushing open-source software stacks around their hardware, each with the ambition of owning the ecosystem or ensuring nobody does, and the effect either way is that the infrastructure layer is opening up. Sovereignty becomes a placement option on that substrate rather than a blocker to work around.

Though the data does not reflect this new development yet. What has arrived is the decision, and the criterion to decide on is not capability but who carries the operational burden of the layer everything else will run on.

This phenomenon is explored in greater detail in The future of AI-native is open source.

The team changed shape to carry it

43% of platform engineers expect their role to shift toward governance and guardrails, more than any other answer. 36% expect to shift from building platforms for humans to building platforms for agents. 11% expect no change, and 9% thinks their role gets automated away.

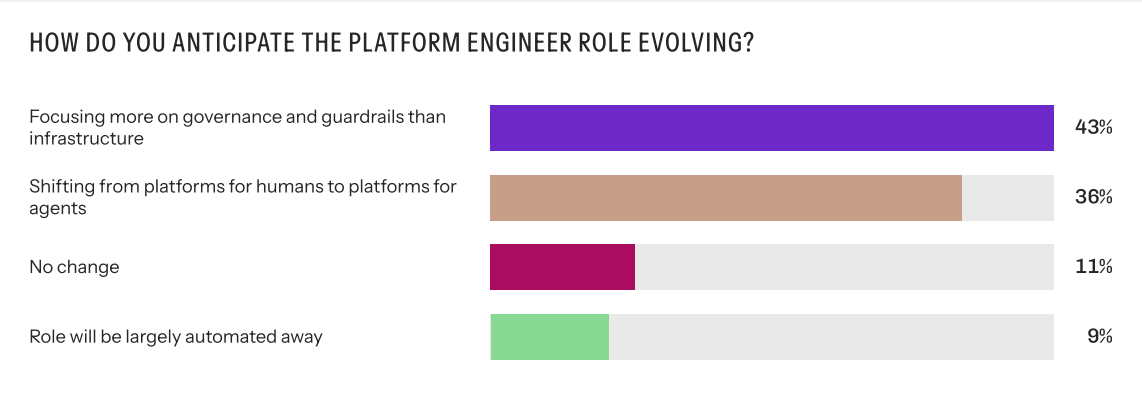


Figure 21: How the platform engineer role is expected to evolve.

Read those together with the security finding from the last chapter. Security handed the mandate over, platform engineering picked it up, and the discipline continued to grow.

A crucial detail to remember: Every specialization Volume 1 predicted has arrived. Data platform engineering, AI platform engineering, AI security, and developer experience for AI are distinct roles now rather than forecasts, and that is the clearest evidence in this survey that the discipline is deepening rather than being absorbed. The team did not get bigger so much as deeper.

The unfinished piece is convergence with data teams. 28% still describe separate silos, and only 6% have fully converged into a single team. Watch the 10% who say they are actively converging rather than the 6% who have arrived, because that is the population that moves the endpoint next year.

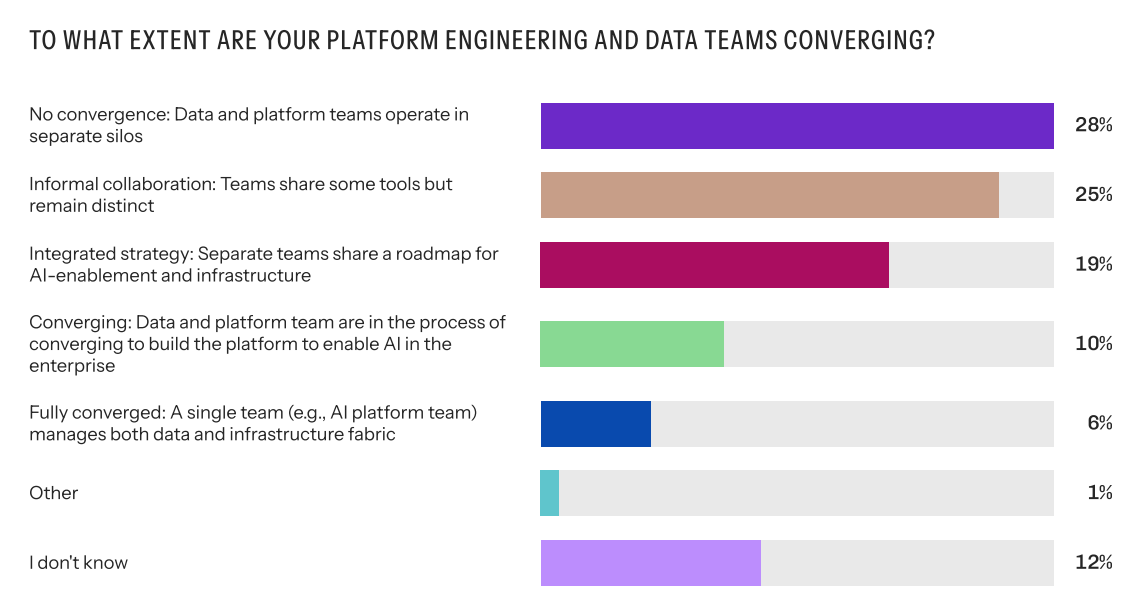


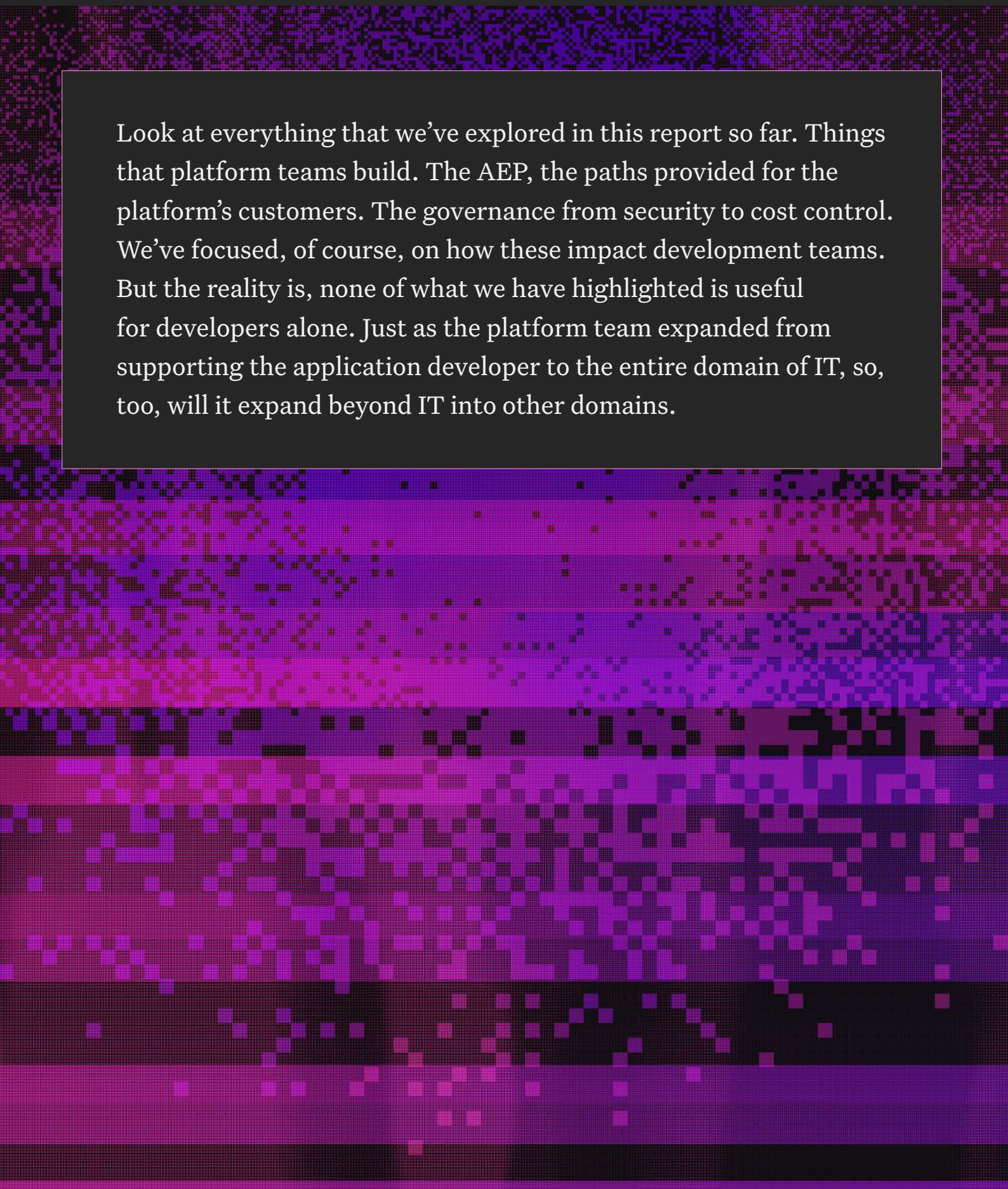
Figure 22: Convergence of platform engineering and data teams

Ownership, though, has consolidated decisively. Nearly half of platform teams provide central model connectivity, extending the pattern Volume 1 found when platform engineering was already the largest single owner of AI infrastructure at 36.7%. Whoever builds the AI base platform becomes the layer the enterprise runs on.

Which is the answer to those 9% who fear their role will be automated away. The function that absorbs recurring work and standardizes it becomes more valuable, not less, as generation gets cheaper, because somebody has to define what correct looks like and somebody has to own the substrate that enforces it. Cheap generation raises the value of that job. It doesn't remove you. You, the platform team, are more important than ever. And that is only going to continue.

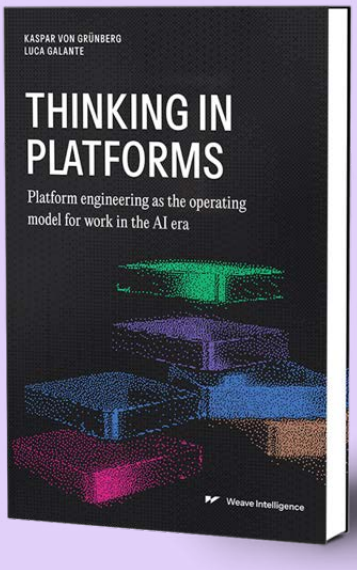
Beyond IT

Look at everything that we've explored in this report so far. Things that platform teams build. The AEP, the paths provided for the platform's customers. The governance from security to cost control. We've focused, of course, on how these impact development teams. But the reality is, none of what we have highlighted is useful for developers alone. Just as the platform team expanded from supporting the application developer to the entire domain of IT, so, too, will it expand beyond IT into other domains.



Take two organizations that were in the same position eighteen months ago. One treated agents as a per-team tooling problem and let every team solve identity, cost, and validation for itself. The other built shared substrate and made every team run on it. The first has a collection of local optimizations that do not compose. The second has something that gets better every time a path is added to it.

Thinking in Platforms, by Kaspar von Grünberg and Luca Galante, takes platform engineering back to its foundations, sets out the Paths to Outcome model, and then hands over the playbook itself, with examples spanning healthcare, marketing, and of course software. Its premise is the one this report keeps arriving at: AI amplifies the system you already have, and the platform is the prerequisite. That fact holds in every domain and every industry, which makes it the starting point for any leader now being asked to enable AI, and build their platform.



The reason the second organization's advantage travels is buried in the architecture. Identity does not know whether the workload is a deployment or a claim approval for the legal team. Cost attribution does not know whether the tokens were spent writing tests or drafting a policy summary. Execution, evaluation, and observability are the same machinery underneath either one. The domain itself lives in the path and in the tooling above it, but not in the agent infrastructure below.

The substrate built by an engineering organization for its agents serves, without modification, as the substrate needed by every other function. To speak directly, the agent infra you build to enable agentic development is the same agentic infra that would be required to enable agentic marketing. What swaps per function are the tooling and the paths; those are built by people who understand that function's recurring work. A hospital, a retailer, or an insurer can climb this entire ladder without ever calling itself a technology company.

Engineering has gone first more for structural reasons rather than cultural ones. The work is text-based. The value stream is well defined. Validation is objective, because tests either pass or they do not. And the unit of work is a legible, reversible object. No other function has all four yet, which is what makes engineering the proving ground and why the team that builds it holds the blueprint for everywhere else. Just as last year, we correctly predicted that the platform engineer would become the AI platform engineer, or the data platform engineer, or the security platform engineer. So, too, now we see the platform engineer becoming the marketing platform engineer or the legal platform engineer. Though they might be called something else, that is the work that they'll be doing, and it's the principles of platform engineering that they'll be using.

None of this is a new argument for the discipline. Platform engineering earned the right to own developer experience by industrializing recurring work once already, and the method it used is indifferent to which function's work is being industrialized. That is the whole claim. Not that platform engineering will expand because it has expanded before, but that the discipline was never specific to software in the first place.

Luca Galante

LEAD ANALYST, WEAVE INTELLIGENCE

Five recommendations to achieve agentic ROI

These are recommendations based on the data, expert interviews, and our own experience working across hundreds of platform engineering organizations at every level of maturity across all industries. Crucially, start with where you actually are, because every recommendation below changes depending on the answer. Locate yourself on the level you operate at, and how much of your lifecycle has governed paths.

LOCATE YOURSELF

Six questions

For each stage below, does a governed, machine-executable path exist that a human or an agent can invoke, with a defined outcome and a way to verify it?

Plan · Code · Review · Test · Deploy · Monitor

Count the yeses. That number is your coverage, and it predicts what your agents will actually deliver better than your level does. Zero to two means the substrate is the work. Three to four means you are in the expensive middle and the gaps are where your throughput is disappearing. Five to six and the constraint has already moved to cost and architecture.

01 Build the four Level 3 prerequisites together, not one at a time

The dispatch path, the validation loop, scoped agent identity, and sandboxed environments only pay off as a set. They must be funded as one program rather than four separate roadmap items, as they are what carry an organization from Level 2 to Level 3, where the returns actually land. 60% of Level 3 organizations report at least doubled throughput, against 36% at Level 2 and 30% at Level 1, and the share reporting a transformative return more than doubles again. Built incrementally, the same four items leave an organization stalled at Level 2, which this report shows is the most expensive place to get stuck.

02 Close the validation loop before adding agent capacity

Route every failure back to the agent with the reason attached, and widen deterministic verification past unit tests into scenario execution, dependency and license analysis, and policy checks on infrastructure changes. Adding generation in front of a human gate does not increase delivery. It only lengthens the queue.

03 Fix attribution now

No borrowed human credentials, no shared service accounts. Scoped identity per agent is the entry ticket to autonomy, and without it, no downstream control can tell an agent's action from a person's. This is mandatory not just to prevent appearing in the news, but also to unlock the power of agents in a massive way.

04 Gate by consequence, not by category

Two questions decide what needs a deterministic gate from day one. They are how much damage the action can do, and how fast you would find out. You should gate anything where the answers are a lot and too late. That is a wider set than the things which cannot literally be undone. It doesn't matter if you have the power to roll back, if the damage happens before you can notice. Everywhere else, deploy visibility into agent behavior, model usage, and cost first, let failures surface in staging and test where they are cheap, and keep detection running in production for what cannot wait. Things like cost overruns, unusual data access, outages. Then restrict. Every rule you write will target a failure pattern you actuallv observed.

05 Measure value not output

Put token spend on a governed budget with per-team and per-path attribution; measure cost per accepted output rather than cost per token. This will control cost. And then measure value in the six vectors rather than in ticket counts, and per the example of the leading agentic enterprises, ROI will become not just something attainable, but actually measurable and reportable too.

What comes next

The change from Volume 1 of this report to this one has been nothing short of supersonic. Even the most sci/fi feeling of predictions have come true, and matured into concrete and actionable fixtures faster than we would have thought possible. We believe that will only continue; however, it will feel less dramatic in the coming 12 months. Moving from the abstract and experimental jagged frontier to the structured construction plan we have now is a very visible and mesmerizing advancement. The change in the next 12 months will instead be about going deeper. Teams will progress up their maturity ladder. The number of agentic enterprises with governed agents delivering transformational ROI will increase. But the fundamental truth will not change. It is the agentic engineering platform, not the model, that delivers the value of AI.

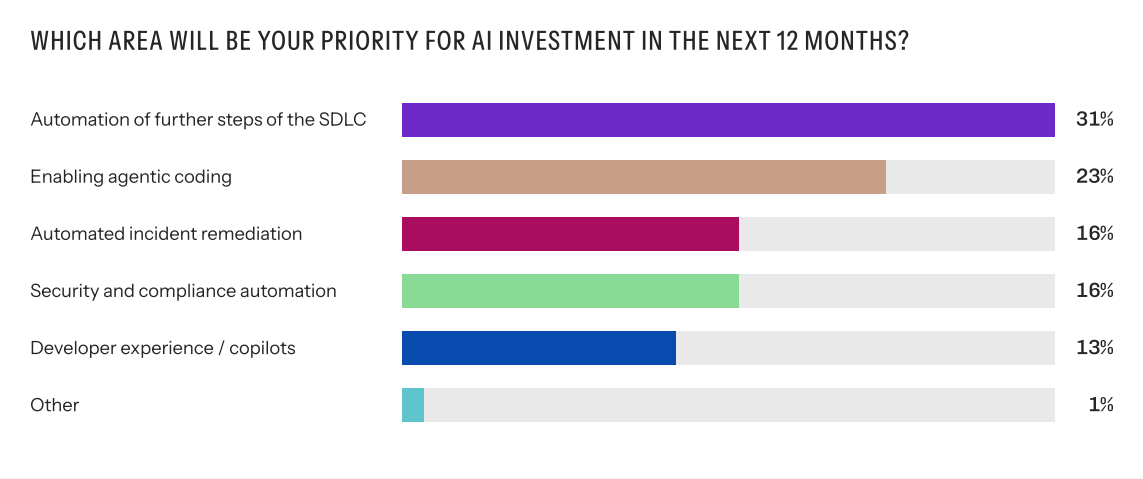


Figure 23: Priority for AI investment in the next 12 months

When asked where they will invest over the next twelve months, practitioners confirm this. 31% put automation of the software lifecycle first, then enabling agentic coding at 23%. Automated incident remediation and security and compliance automation tie for third at 16% each, with developer experience and copilots last at 13%. The field intends to build precisely the capabilities this report argues are the constraint. The industry already knows what is wrong, and is racing to solve it.

By next year, we expect more than 30% of organizations to report Level 3 or above, up from 18% today, on the assumption that validation loops become a standard platform offering rather than a bespoke build in every organization. That is deliberately slower than the Level 1 to Level 2 wave of the past year, because the step to Level 3 is gated by platform capability rather than by tool adoption, and only 17% of organizations run the loops it requires today.

We also assert that Level 3 will become the norm for forward-leaning teams. Agent-specific compliance and attribution frameworks will emerge, driven by the auditability and residency concerns already sitting at 43% each. Regulated sectors will move toward sovereign and self-hosted stacks growing from the currently low base of 3% and 11%. New waves of agents and agent-to-agent patterns will arrive. Platform teams will build agentic platforms spread function by function, with the agent infrastructure layer traveling first, where the durable advantage for an enterprise will turn out to be neither any single platform nor any single function's autonomy, but the cross-functional loop that connects them with shared gates and shared state.

Take these predictions and revisit the core numbers at the beginning of this report. 38% of organizations ship at least twice as much since adopting AI, yet only 8% call the return transformative. Every chapter in between was an attempt to close that gap, and each one found a different piece of why the gap exists. Absorption explained why more output doesn't become more delivery. ROI explained why delivery doesn't automatically become money. FinOps explained why so much of that spend goes unmeasured. Governance explained why, for more than half the industry, nobody can even say for certain whether a person or an agent did the work. These are four separate failures, and the same platform closes all four, because all four are the same problem wearing different clothes. That is the throughline this whole report has been arguing for from chapter one. Though we cannot predict specific increases in ROI or throughput across the industry, they will come in the next 12 months, as they will be a natural outcome of platform work that is being done.

The lifecycle may have changed shape, but the platform has evolved to meet it. That is the whole story of the past twelve months. The organizations that built to match the new shape are being paid for their agents. The industry's own ranking of its obstacles settles where the differentiator sits with platform readiness the number one obstacle, and model capability only fourth. Nobody serious is waiting for a better model anymore.

A year ago this report described an industry that had adopted AI everywhere and had almost nothing to show for it. Volume 1 was written to explain that gap. This report is written to say the gap is not permanent, and it is not mysterious. It closes for the organizations that stop treating the platform as something agents pass through, and start treating it as the thing that decides what agents are allowed to become. Everyone in this industry now has access to the same models. What they build around them is the only thing left to compete on.

Appendix

Survey methodology snapshot

The insights presented derive from comprehensive primary research:

- **Participants**: 350 professionals globally.

- **Roles**: Platform engineers, DevOps engineers, SREs, architects, consultants, and technical leaders, spanning individual contributors (31%) through team leads (27%) and heads of function (19%) to VP, director and C-level (23% combined).

- **Timing**: Data collected April to August 2026

- **Scope**: 24 questions covering agentic maturity, AI usage, throughput and ROI, security and governance, cost management, organizational attitudes, the platform team's mandate, and future expectations.

- **Purpose**: Understanding how far organizations have moved from adopting AI to delivering with agents, and what the platform has to provide for that shift to pay.

Glossary

Agent	A model plus its harness. The model supplies the reasoning; the harness supplies the execution, context, capability and evaluation that turn reasoning into work. Nothing in this report treats the model alone as the agent.
Agent infrastructure	The layer carrying identity, context, capability, execution, evaluation, security and observability for every agent on the platform, defined as code. It is vertical-agnostic, so the same substrate that serves software delivery would serve any other function. Its three building blocks are the harness, governance and the models.
Agentic Engineering Platform (AEP)	What the internal developer platform becomes when agents are users of it rather than tools bolted onto the side of it. It makes platform paths executable by agents at scale, and refers specifically to the development vertical. It adds layers to the IDP rather than replacing it.
Agentic Software Development Lifecycle (ASDLC)	What the SDLC becomes when agents are active participants alongside human engineers. The stages and their order survive; only the actors change, with agents occupying code, review and test while humans hold the gates between them.
AI implementation plateau	The stage where an organization stops seeing rapid gains from AI adoption and meets slower ROI, integration hurdles and the need for deeper cultural or process change. Volume 1 identified it as the reason near-universal AI usage produced so little organizational return.
Deterministic path	A pipeline-driven path that produces the same output for the same input: builds, scans, policy gates, promotions, deployments. Governance is cheap because the behavior is knowable in advance, which makes it the correct default for anything auditable, irreversible or regulated. It runs on the tooling layer with no agent infrastructure beneath it.
Dispatch path	The path that converts a human-directed assignment into a governed agent work item with identity, context and scope attached. It is the first genuinely new path an organization needs at Level 2, and one of the four Level 3 prerequisites.
Enterprise citizen developer	A non-technical business user who builds workflows and applications on top of platform infrastructure through AI-powered interfaces. They arrive as platform customers from Level 3 onwards, and 48% of organizations in this survey already count them as such.

Harness, and harness engineering	The harness is everything around the model that turns intelligence into work: execution, context, capability and evaluation. Harness engineering is the craft of tuning a single agent turn across that machinery. It is a practice within platform engineering rather than a replacement for it, because the harness runs one agent while the platform governs the fleet.
Hybrid path	Probabilistic generation wrapped in deterministic gates, looping until a gate passes or the budget is spent. Higher token cost per unit of work, full auditability at the gate, and blast radius bounded by whatever the gate refuses to let through.
Internal Developer Platform (IDP)	A platform built by a platform team to standardize paths for developers, enabling self-service and reducing cognitive load. In this report it is the deterministic frame agents operate inside, and the foundation an AEP builds on rather than legacy to be replaced.
Levels of agentic development	The framework classifying maturity by how much work an agent is trusted to carry and where the human sits relative to the loop. Level 0, human is the loop, has no agents in the value stream. Level 1 is human in the loop, Level 2 human on the loop, Level 3 humans as orchestrators, and Level 4 systems of agents initiating and completing work inside human-designed guardrails.
Loop engineering	The craft of tuning what happens between agent turns: the gate verdict, the feedback that goes back, the retry and the stopping condition.
Path	A repeatable route from user intent to usable output, and the fundamental building block of any platform.
Probabilistic path	An agent-driven path that returns a plausible output rather than a guaranteed one: an agent reviewing a change, drafting a specification, triaging an alert. Correctness cannot be established by inspecting the path definition, so the governance cost moves into the evaluation apparatus. Evals become the test suite.
Shadow AI	Unmanaged AI tools and accounts used inside an organization without central approval or visibility. 27% of organizations in this survey call it significant, and 8% have no visibility into individual AI tool usage at all.

Tokenomics	The cost dimension of running agents at scale. Every retry consumes tokens, so token spend prices the attempt rather than the outcome, which is what makes cost per accepted output the number a platform team can actually improve.
Tooling layer, and the five platform planes	The top layer of an AEP, made up of the five planes platform teams already run: developer control, integration and delivery, resource, security, and observability. Under agent-scale demand what changes is load rather than structure, which turns reliable APIs, predictable rate limits, structured outputs and explicit failure modes from good practice into preconditions.
Validation gate	A single-pass checkpoint a change either clears or fails, with approval as a binary outcome. Deterministic gates check hard criteria such as tests, security scans and policy; probabilistic gates use a model as judge against softer criteria such as intent and architectural alignment.
Validation loop	Validation as an industrial feedback mechanism rather than a checkpoint. When a check fails the failure routes back to the agent with the reason attached, the agent revises and retries, and the cycle repeats until the deterministic checks pass. First-pass failure is convergence rather than defect, and this is the mechanism that lifts the review-bandwidth cap.

Further reading

1. Sam Barlien and Luca Galante, State of AI in Platform Engineering 2025 Volume 1 (Weave Intelligence, 2025)

2. Kaspar von Grünberg, Luca Galante, Ajay Chankramath, and Mallory Haigh, The four levels of agentic software development in the enterprise (Weave Intelligence, 2026)

3. Sam Barlien, "Operationalizing AI coding agents in regulated industries," Weave Intelligence, 2026.

4. Kaspar von Grünberg, "From IDP to AEP: Why platform engineers now build agentic development platforms," Weave Intelligence, June 2026.

5. Kaspar von Grünberg, "Stop wasting tokens. Build a platform," Weave Intelligence, June 2026.

6. Kaspar von Grünberg, "What is an Agentic Engineering Platform (AEP)?," Weave Intelligence, May 2026.

Disclaimer

Weave Intelligence does not endorse any specific vendor, product, or service. The information contained in this report has been obtained from sources believed to be reliable. Weave Intelligence disclaims all warranties as to the accuracy, completeness, or adequacy of such information. This publication is provided on an “as-is” basis without warranty of any kind, either express or implied. Weave Intelligence shall have no liability for errors, omissions, or inadequacies in the information contained herein or for interpretations thereof. The reader assumes sole responsibility for the selection of these materials to achieve its intended results.

About Weave Intelligence

Weave Intelligence

Weave Intelligence is a leading analyst firm specializing in platform engineering. By uniting a team of senior analysts with industry experts and enterprise leaders, we deliver the rigorous research that defines the field. We enable organizations to leverage the #1 trend in IT as the modern framework for operational excellence and innovation.

weaveintelligence.io
research@weaveintelligence.io

Weave Intelligence GmbH
Wöhlertstraße 12-13
10115 Berlin