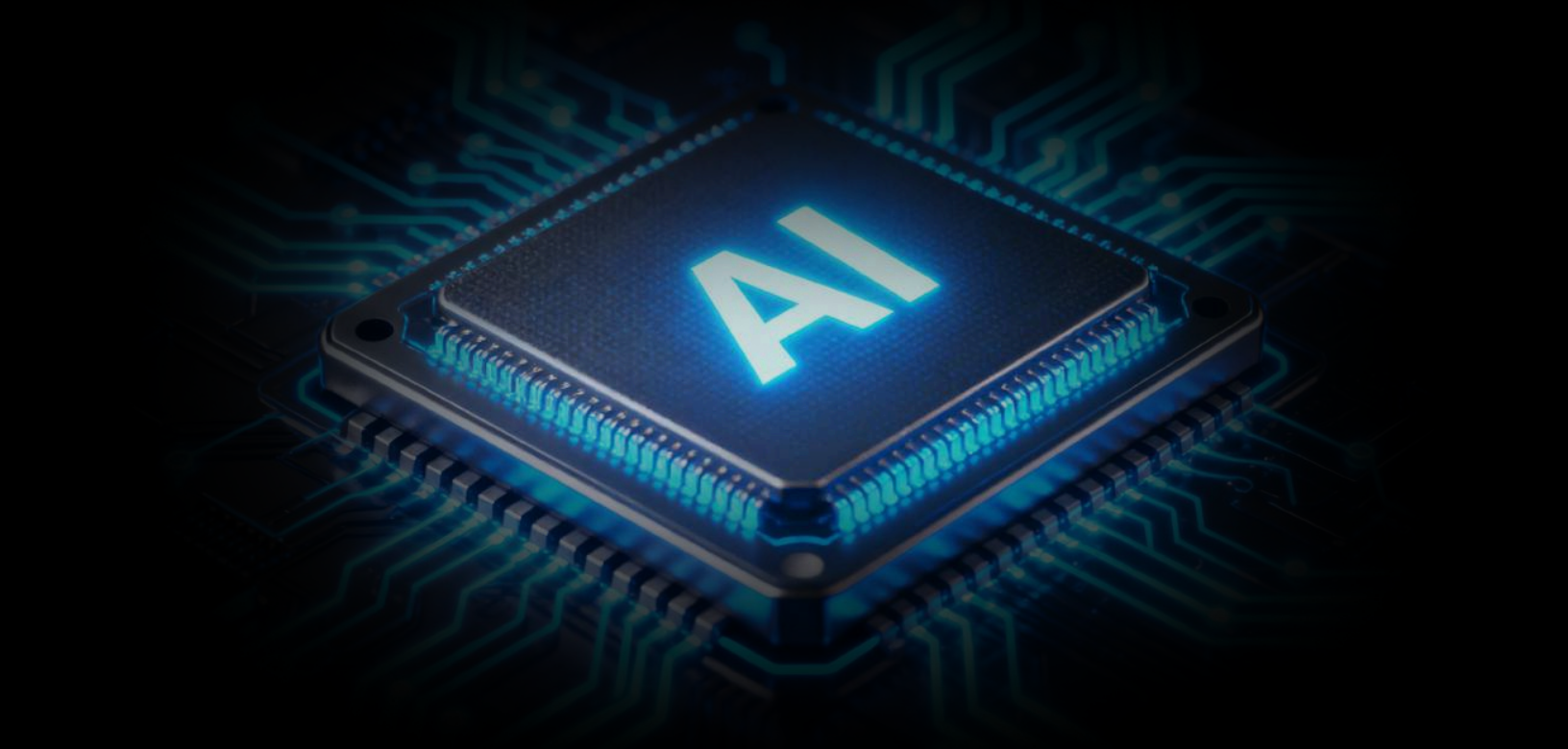
harness

The AI ROI Playbook: From Token Spend to Business Outcomes

94% of engineering leaders say the metrics that matter most to measure AI ROI are missing.

Learn how to build frameworks that connect AI tool investment to shipped software, customer value, and bottom-line results.



Contents

Executive Summary

Global AI software spending is expected to reach $2.59 trillion in 2026, a 47% year-over-year increase, according to Gartner. And yet, according to the Harness 2026 State of Engineering Excellence report, 94% of engineering leaders say the metrics that matter most are missing from their current measurement frameworks. The executives closest to the problem are already naming it out loud:

> **"I am going to probably use $300 million of Anthropic tokens this year at Salesforce. Coding."**
>
> — Marc Benioff, CEO, Salesforce

> **"There's a bill to pay for those tokens. It's almost like measuring a restaurant based on how many ingredients they buy."**
>
> — Chris Bedi, CCO, ServiceNow

The problem has two sides and most organizations are failing on both. On the developer side, engineers are using AI to write nearly every line of new code, but leaders have no visibility into whether that token spend is generating software that ships. On the infrastructure side, production agents consume tokens with every customer interaction, and the monthly invoice is the only signal on whether any of it is worth the cost.

Most organizations treat these as separate problems owned by separate teams. FinOps worries about the infrastructure bill, while engineering leadership worries about developer productivity. In practice, they are the same investment question. Organizations that keep the conversations siloed will keep answering only half of it.

This playbook provides the frameworks to close both gaps simultaneously: visibility across all AI spend, metrics that connect spend to outcomes, and governance that works before cost surprises reach the board. Chapter 1 explains why current frameworks fail for AI spend. Chapter 2 defines the two measurement gaps and the signals required to close each. Chapter 3 maps AI spend across three layers with distinct cost drivers and ROI metrics. Chapter 4 builds the measurement frameworks for both sides. Chapter 5 covers governance, a 90-day implementation plan, and a maturity scorecard.

The formalization of this challenge is already underway at the industry level. In June 2026, the Linux Foundation launched the Tokenomics Foundation, with JPMorgan Chase, Salesforce, Accenture, Booking.com, AWS, and Microsoft as founding members. The Tokenomics Foundation exists to create vendor-neutral standards and governance practices for managing enterprise AI token costs. Its launch signals that tokenomics, the discipline of measuring and attributing AI token spend to business outcomes, is no longer a concept. It is an institutionalized practice. This playbook operationalizes the consumption layer of that discipline: connecting the tokens your teams and agents consume today to the outcomes your business needs to measure tomorrow.

This playbook is for engineering leaders accountable for AI tool ROI, FinOps and finance leaders governing AI infrastructure spend, and C-suite executives evaluating AI investment at the board level.

CHAPTER 1

Why the Current Frameworks Don't Work

Traditional engineering metrics weren't built for AI

Traditional engineering metrics were built for a world where software development began at the commit. A developer sat down, wrote code, opened a pull request, and the measurement infrastructure — cycle time, PR throughput, deployment frequency — picked up from there. Everything that happened before the commit was invisible by design, because there was nothing to measure.

AI changed where software development begins. Today, a developer might spend 40 minutes in a conversation with an AI coding assistant before writing a single line of code directly. They may accept, reject, and iterate on dozens of AI suggestions before a PR is ever opened. The work that used to start at the keyboard now starts at the prompt. And the metrics built around the commit boundary capture none of it.

This isn't a minor gap. Two developers with identical PR counts and cycle times might have radically different AI usage patterns. One shipping high-quality, AI-assisted code efficiently, the other generating and discarding tokens at scale without the output to show for it. Traditional DORA metrics and delivery dashboards treat them identically. They were never designed to see the difference.

Closing this gap requires tracking the full development lifecycle, from the moment AI assistance begins to the moment code reaches production. That means connecting AI coding activity to the downstream pipeline: pull requests opened, reviews passed, code merged, deployments shipped. Only when those two data streams are joined can you see which AI-assisted work actually created value and which generated tokens without output.

> **For engineering leaders:** Adoption metrics were the right starting point. They're the wrong ending point. If your current AI tooling reports tell you how many suggestions were accepted but not how many of those suggestions shipped to production, you're measuring the input, not the outcome. The shift to outcome-based measurement is what separates organizations that can prove AI ROI from those that are still estimating it.

Traditional FinOps was built for cloud infrastructure, where the playbook was straightforward: tag resources, allocate to teams, set budgets, and monitor utilization. The model worked because costs were predictable at the resource level.

AI workloads break every assumption in that model and they do it on both the developer and infrastructure sides simultaneously.

The developer visibility gap

Developer AI tools like Copilot, Cursor, and Claude Code are typically purchased as per-seat SaaS subscriptions managed by engineering or IT, which are invisible to FinOps. Their impact on velocity, code quality, and delivery throughput has never been systematically measured. Most organizations know their seat count. Almost none know whether those seats are producing shipped software or just generating tokens.

The infrastructure attribution gap

Production AI agents generate invoices from OpenAI, Anthropic, AWS Bedrock, or GCP Vertex AI with no attribution to the team, feature, or customer interaction that drove the cost. A single line item on a monthly bill might represent a high-ROI support agent resolving thousands of tickets cheaply or an agent consuming tokens without producing outcomes. The invoice alone cannot tell you which.

This creates three structural gaps no traditional FinOps tool was designed to fill:

- **Attribution:** Which team, feature, or customer generated this spend and on both the developer tool and infrastructure sides?
- **Efficiency:** Are tokens producing value, or being wasted on unshipped code and failed agent runs?
- **Unit economics:** What is the cost per outcome, per PR merged, per ticket resolved, per feature shipped?

Key insight: The Tokenomics Foundation, launched at FinOps X 2026, was established to create vendor-neutral standards for managing enterprise AI token costs, filling the governance gap this chapter describes. This playbook operationalizes its consumption layer: connecting token spend to the outcomes your business tracks.

One invoice and four stakeholders who need different answers

Stakeholder	What they're asking	What a good answer requires
CTO	"Is AI changing how fast we build and ship?"	Delivery throughput delta. Evidence that AI investment is moving the engineering capability curve.
VP Engineering	"Are AI tools making my team faster?"	Delivery impact per dollar. Token spend traced to shipped work, per team.
CFO / Finance	"I can't budget a line that swings 10x."	Budgets, guardrails, anomaly detection. AI governed like cloud spend.
CEO / Board	"Is this dollar coming back as more?"	Spend tied to outcomes. Unit economics the board can evaluate.

For FinOps practitioners: Your entry point to AI cost governance is the same one you used for cloud. However, the tagging taxonomy, attribution model, and anomaly thresholds all need to be rebuilt from scratch. The FinOps Foundation's AI working group is a useful starting point. The frameworks in this playbook give you the operational layer.

A complete framework has to serve all four perspectives from the same underlying data. That means connecting developer token spend to delivery outcomes and connecting infrastructure AI spend to business outcomes simultaneously, not sequentially.

CHAPTER 2

The Two Measurement Gaps

AI cost management has two distinct measurement gaps. One on the developer side and one on the infrastructure side. Organizations that solve only one are still flying partially blind. The frameworks for each are different, but the underlying principle is the same: connect spend to outcomes, not spend to usage.

Gap 1: Developer spend disconnected from delivery

The first wave of AI deployment had one goal: adoption. Seat counts, active users, and token consumption were the metrics. That phase is over. The question has shifted from "are people using it?" to "is it working?" And this requires a completely different measurement infrastructure.

Engineers understand the term that captures the failure mode precisely: **tokenmaxxing.** An engineer burns 500,000 tokens generating code that gets rejected in review. By the adoption leaderboard, they beat the engineer who shipped a clean 50-line patch straight to production. The metric rewards consumption. It doesn't reward outcomes.

Tokenmaxxing is not a developer behavior problem. It's a measurement problem. Developers optimize for what gets measured. The organizations still running token consumption leaderboards are optimizing for the wrong signal.

Closing this gap requires four signals most organizations are collecting in separate places, or not collecting at all:

- **Ship rate:** Of the code AI generates and a developer accepts, what fraction reaches production? Low ship rate is the clearest tokenmaxxing signal.
- **Cost per merged PR:** What does it cost in AI credits to produce a merged, production-ready pull request? This reframes the conversation from monthly seat cost to cost per unit of delivered work.
- **Wasted spend:** What fraction of tokens generates code that never ships, which includes abandoned drafts, rejected suggestions, failed reviews? Above 15% signals prompts or habits that need attention.
- **Delivery throughput delta:** Are AI-assisted developers shipping more frequently than non-AI peers on the same team? This is the ROI signal that connects tool spend to engineering output.

Organizations that have built this visibility report results that reframe the entire AI investment conversation — from adoption stats to delivery metrics. Based on Harness internal developer data:

+149%	+47%	+77%	-32%
PR velocity per engineer	features shipped per engineer	bug resolution rate improvement	rework rate reduction

For VP Engineering and engineering managers: Ship rate and cost per merged PR are the two numbers to bring into your next AI tool review. If your team lead can tell you acceptance rate but not what percentage of accepted code shipped to production last quarter, the measurement infrastructure isn't there yet. These are the metrics that let you have a budget conversation, not just an adoption conversation.

For FinOps practitioners: Developer tool spend is the blind spot in your current model. Seat-license costs flow through IT or engineering budgets, outside normal FinOps visibility. The first step is inventory: how many seats across which tools, at what monthly cost. The second step is connecting that cost to delivery output using the ship rate and cost-per-PR metrics described here.

Gap 2: Infrastructure spend disconnected from outcomes

Once an AI agent ships to production, a different cost problem takes over. Every customer interaction, every resolved ticket, every automated workflow triggers inference. The spend scales with usage and in most organizations is visible only at the invoice level that shows which line item is growing, but nothing about whether that growth is justified.

This is a structural attribution failure, not just a reporting gap. A support agent, a code review assistant, and a content generation pipeline may all appear as a single line on a provider invoice. Without tagging and tracing at the agent and session level, cost optimization is impossible.

Addressing this disparity requires attribution at a level most organizations don't yet have. A $28,000 monthly agent bill needs to be traceable not just to a provider, but to the agent, the session, the model call, and the individual span because that is the level where optimization is possible. Once you can see that 72% of a support ticket's cost is the model call for generating the reply (based on Harness CACM data), you know where to focus. Until then, you're trying to optimize a number you can only see once a month.

The FinOps signals that matter on the infrastructure side mirror the developer signals in principle:

- **Cost per outcome:** Cost per resolved ticket, cost per conversion, and cost per generated asset. The unit that maps to business value for each feature.
- **Agent efficiency rate:** What fraction of agent runs complete successfully versus failing, retrying, or producing outputs that require human correction?
- **Spend growth vs. outcome growth:** Is monthly agent cost growing faster or slower than the business value it's producing? Divergence is the early warning signal.
- **Anomaly detection per agent:** Cost spikes caught at the agent level, not the provider level, so a misbehaving prompt or model change is flagged before it compounds across a billing cycle.

The infrastructure equivalent of tokenmaxxing is throughput theater: dashboards that show agents running, requests processing, and invoices growing, with no signal on whether any outcome was produced. Your monitoring shows uptime. Your performance dashboard shows latency. Your FinOps tool shows spend. None of them show whether the customer got their answer, the ticket got resolved, or the feature shipped. On the developer side, teams are tokenmaxxing. On the infrastructure side, they are running throughput theater. Both are the same failure with a different face: measuring the wrong thing and calling it accountability.

For product and engineering leaders: Throughput theater is a resource allocation problem disguised as a performance problem. If your agents are running, your dashboard looks healthy, and your invoice is growing — but you cannot tell whether the outcomes justify the cost — the issue is not your agents. It is the absence of a cost-per-outcome baseline to measure them against. Define that unit before the next planning cycle, not after.

For FinOps practitioners: Infrastructure AI is where your existing cloud governance skills transfer most directly including tagging, allocation, anomaly detection, and budget governance. The key difference: AI costs move faster and the attribution model is more complex. A single model upgrade or prompt change can double spend overnight. Monthly reviews are a floor, not a cadence.

CHAPTER 3

A Taxonomy of AI Spend

Before building measurement frameworks, you need a clear map of where AI spend lives in your organization. Most teams discover it's spread across three layers, each with different cost drivers, attribution models, and ROI metrics. Treating them as one budget line is the root cause of most AI ROI failures.

Layer 1: Developer AI Tools

Seat-license tools (GitHub Copilot, Cursor, Claude Code, Windsurf) owned by engineering or IT, invisible to FinOps. The tokenmaxxing problem lives here and without delivery tracing, the only available signal is consumption.

ROI question: Is AI-assisted code reaching production? Are developers shipping more, with fewer defects, in less time or just generating more tokens?

Layer 2: AI-Powered Product Features

The inference layer supporting agents, recommendation engines, search, and generative features. This layer scales directly with product usage, making it the most volatile layer and the highest FinOps priority.

ROI question: Is the feature worth more than it costs to run? Is the feature worth more than it costs to run? You need a unit that maps to business value (cost per resolved ticket, cost per conversion) and the discipline to govern spending against it.

Layer 3: AI Infrastructure

The training and fine-tuning layer, like GPU compute, vector databases, and embedding pipelines. These are highly bursty: one training run can generate more cost in 48 hours than a month of inference.

ROI question: Is building proprietary capability worth the cost versus a managed API? This is fundamentally a make-vs-buy question. Evaluate it by measuring the performance delta your fine-tuned model delivers over the base API, the time-to-capability advantage it creates, and the total cost over a 12-month horizon, not just the training run. In most cases, managed APIs win until the performance gap justifies the ongoing training and maintenance cost.

Layer	Developer Tools	Product Features	AI Infrastructure
Primary ROI metric	Ship rate, cost per merged PR, wasted spend, cycle time delta	Cost per outcome — ticket, conversion, asset, interaction	Model performance delta per training dollar, time-to-capability

For CFO and finance leaders: Treat these three layers as distinct budget items with unique governance. Budget developer tool spend per seat based on delivery output; product feature spend against cost-per-outcome ceilings; and infrastructure spend by project milestones. Grouping these into one line item prevents effective governance and forecasting.

CHAPTER 4

Building Measurement Frameworks That Work

With the taxonomy clear, the challenge is building the right measurement infrastructure for each layer. Both frameworks follow the same principle: connect spend to outcomes, not spend to usage. The metrics are different; the discipline is the same.

Developer AI: from acceptance rate to ship rate

Code acceptance rate is the most commonly tracked developer AI metric. It tells you whether developers are keeping suggestions. It does not tell you whether those suggestions are shipping, improving quality, or reducing cycle time. Most AI ROI claims fall apart in the gap between acceptance rate and delivery outcome.

Metric	Formula	What it tells you
Code acceptance rate	Accepted / total suggestions	Whether developers find AI suggestions useful; a leading indicator, not an ROI metric
Ship rate	AI code in production / AI code accepted	Whether accepted code creates value or feeds the tokenmaxxing problem
Cost per merged PR	Monthly AI tool cost / PRs merged	Unit cost of AI-assisted delivery (the number leadership can evaluate)
Wasted spend	Tokens on unshipped code / total	The tokenmaxxing signal; high waste means spend without output
Cycle time delta	AI-assisted days per PR vs. baseline	The delivery speed return on AI investment

What this means for leadership

For CTOs and VPs of Engineering: The single most important shift is demanding ship rate alongside acceptance rate in every AI tool review. If your vendor's dashboard shows acceptance rate but not what percentage of accepted code reached production, you are measuring activity, not output.

For CFOs: Cost per merged PR is the number to benchmark and govern. It converts AI tool spend into a unit cost directly comparable to other engineering investments — and it is the number that makes the renewal conversation substantive rather than anecdotal.

Infrastructure AI: from invoice to cost-per-outcome

The right unit for evaluating production AI spend is cost per meaningful outcome, not the monthly provider total.

For every feature, this calculation has three inputs: total AI cost (token consumption × per-token price plus overhead), total successful outcomes (tickets resolved, not just handled; conversions, not just clicks), and the baseline cost of the same outcome without AI.

Infrastructure AI metric	Formula	What it tells you
Cost per outcome	Total AI cost / successful outcomes	Whether the feature creates value relative to what it costs
Agent efficiency rate	Successful runs / total runs	What fraction of spend is producing usable output vs. being wasted on retries and failures
Spend-to-outcome growth ratio	Month-over-month cost growth / outcome growth	Whether the investment is scaling efficiently or degrading
Per-agent anomaly rate	Cost variance per agent vs. baseline	Early warning signal for model changes, prompt regressions, or runaway sessions
Human cost delta	Human cost per outcome / AI cost per outcome	The ROI case in a single number

A $28,000 monthly spend on a customer support agent is a completely different number depending on how many tickets it resolved. If it costs $0.60 per resolved ticket against a $4.50 human alternative, it is one of the best investments in your stack. If the math runs the other way, you are paying more for automation than the process it replaced.

AI feature type	Recommended unit economics metric
Customer support agent	Cost per resolved ticket vs. human agent cost
Code review assistant	Cost per actionable comment vs. senior engineer time
Content generation	Cost per published asset vs. human writing/design cost
Search / recommendations	Cost per conversion vs. baseline conversion rate
Onboarding / documentation	Cost per user activation vs. human onboarding cost

What this means for leadership

For CEOs and board members: Shift focus from total infrastructure spend to cost per resolved outcome. Comparing AI costs to traditional alternatives transforms AI from an expense into a strategic investment.

For CFOs: Mandate cost-per-outcome thresholds for all production AI. Like untagged cloud spend, features without defined outcome units or baseline comparisons should remain unbudgeted until measurable.

CHAPTER 5

Governance and the 90-Day Plan

Measurement without governance is just reporting. The goal is not to know how much you're spending on AI. It's to own that spend, attribute it correctly, and intervene before problems become invoices. This applies equally to developer tool spend and infrastructure spend.

Governance fundamentals

AI cost ownership is contested by default. It touches FinOps, Finance, engineering, product teams, and AI Centers of Excellence simultaneously. The fix is not a new team. It's extending existing ownership with explicit responsibility:

- **FinOps:** billing normalization across all AI providers, budget governance, anomaly detection per agent, cross-provider cost allocation
- **Engineering leadership:** per-team developer tool ROI, wasted spend targets, ship rate benchmarks, delivery efficiency accountability
- **Product and finance:** unit economics review, cost-per-outcome thresholds, build-vs-buy decisions for new AI features

Two governance rules apply equally to both sides. First, consistent tagging is non-negotiable: every API call, GPU workload, developer tool seat, and production agent must be attributed to a team, product, and cost center before any governance model can function.

Second, AI costs move faster than cloud costs. A single model change or new feature can double monthly spend in days. Weekly reviews are a floor for developer spend; real-time anomaly detection per agent is the standard for infrastructure spend.

The most durable improvement is embedding cost visibility in engineering workflows directly. When cost per pull request and cost per ticket are visible where engineers work, not in a separate finance dashboard, cost becomes an input to daily decisions rather than a monthly surprise.

FinOps operating model for AI: AI governance should extend your existing cloud governance practice, not replace it. Apply the same tagging taxonomy you use for cloud, like team, product, cost center, environment, and add two AI-specific dimensions: model name and agent/feature ID. This enables the per-agent anomaly detection and cost-per-outcome tracking that invoice-level reporting cannot provide. Establish a monthly AI cost review cadence with engineering leads that mirrors your existing cloud spend review, and set anomaly thresholds at the agent level before the next billing cycle closes, not after a spike has already compounded.

For engineering leadership: Governance accountability on the developer side means owning waste rate targets and ship rate benchmarks at the team level — not delegating them to FinOps or finance. The teams that close the AI ROI gap fastest are the ones where engineering leadership treats cost per merged PR the same way they treat deployment frequency: as an engineering metric, not a finance one.

The 90-day plan

Week 1-4

Baseline and tag: Audit spend across all three layers. Apply a standard tag taxonomy to all new AI spend immediately. Establish per-developer token cost and wasted spend baselines. Connect top AI providers to a cost management layer and establish per-agent baselines. Identifying developer licenses with zero active usage (ghost seats) are instant savings with no tradeoffs.

Week 5-8

Instrument and measure: Run a 30-day comparison between AI-assisted and non-AI developers on the same team. Track ship rate, wasted spend, and PR cycle time. For your top production agent, define the unit of value and start tracking cost-per-outcome weekly. Enable anomaly detection per agent before the next billing cycle closes.

Week 9-12

Present and govern: Identify highest-ROI developer usage patterns and share them across teams. Prepare a one-page AI ROI summary — developer delivery impact alongside infrastructure unit economics. Present to engineering and finance using cost-per-outcome framing, not total spend. Set internal price targets for top AI features and make them part of engineering operating metrics.

Executive 90-day checklist: At the end of 90 days, ask your team three questions:

1. What is our cost per merged PR across AI-assisted developers, and how does it compare to 90 days ago?
2. What is our cost per resolved ticket (or equivalent outcome unit) for our top production AI feature?
3. Do we have per-agent anomaly detection active, and what was the most significant cost spike we caught this quarter?

If your team can answer all three, you have measurement infrastructure. If they can't, you have the 90-day plan.

Maturity scorecard

Capability	Developer signal	Infrastructure / FinOps signal
Spend attribution	Token spend traced to developer, tool, and team	All agent spend attributed to provider, team, and use case
Waste visibility	Wasted spend tracked per team — target below 15%	Agent efficiency rate tracked — failed and retry runs flagged
Outcome tracing	AI-generated code tracked from prompt to production	Cost-per-outcome defined for top 3 production features
Unit cost baseline	Cost per merged PR established and tracked monthly	Cost per resolved ticket / conversion established and tracked
Anomaly detection	Wasted spend spikes flagged before renewal	Per-agent cost alerts active, not just per provider
Governance cadence	Monthly developer AI cost review with engineering leads	Monthly infrastructure AI review with FinOps and finance

If you can check all six rows on both sides, you're ahead of 94% of engineering organizations. Start with the first row you can't check on either side.

CONCLUSION

The Measurement Advantage

The first phase of enterprise AI was adoption. That phase is over. The second phase is ROI and it requires a different kind of infrastructure.

The organizations that win the ROI phase are not the ones that spent the most in the first phase. They're the ones building the measurement infrastructure now: ship rate and cost-per-PR on the developer side, cost-per-outcome and per-agent anomaly detection on the infrastructure side.

The tokenmaxxing era and the invoice-only era share the same root cause: measurement infrastructure that was never built to ask the right question. The right question is not “how much did we spend?” It is “what did we produce and what did it cost per unit of value delivered?”

The organizations that answer that question first across both the developer and infrastructure sides will be in a different position when the next round of AI investment decisions gets made. Not just because they spent wisely, but because they can prove it.

Start with one layer. Measure one outcome. The 90-day plan is designed to move the conversation from "we spent X on AI" to "we spent X and produced Y" within a quarter. That shift changes how engineering investment is evaluated, how infrastructure spend is governed, and how the case for continued investment gets made.

How Harness Closes These Gaps

Harness AI DLC Insights closes the developer measurement gap. It connects every AI-generated line of code to the PR, ticket, and deployment it produced, giving engineering leaders ship rate, waste rate, cost per merged PR, and delivery throughput delta across every coding agent, team, and developer.

Harness Cloud & AI Cost Management (CACM) closes the infrastructure measurement gap. It traces every dollar to the agent, session, and model call that generated it across OpenAI, Anthropic, AWS Bedrock, and GCP Vertex AI, with per-agent anomaly detection and cost-per-outcome tracking that catches problems when they happen, not on next month's invoice.

Together, they connect both sides of the AI investment equation in a single platform.

Want to learn more? Request a Demo with Harness.

harness

The AI-Native Software Delivery Platform

Follow us on

/harnessio

/harnessinc

Contact us on

www.harness.io