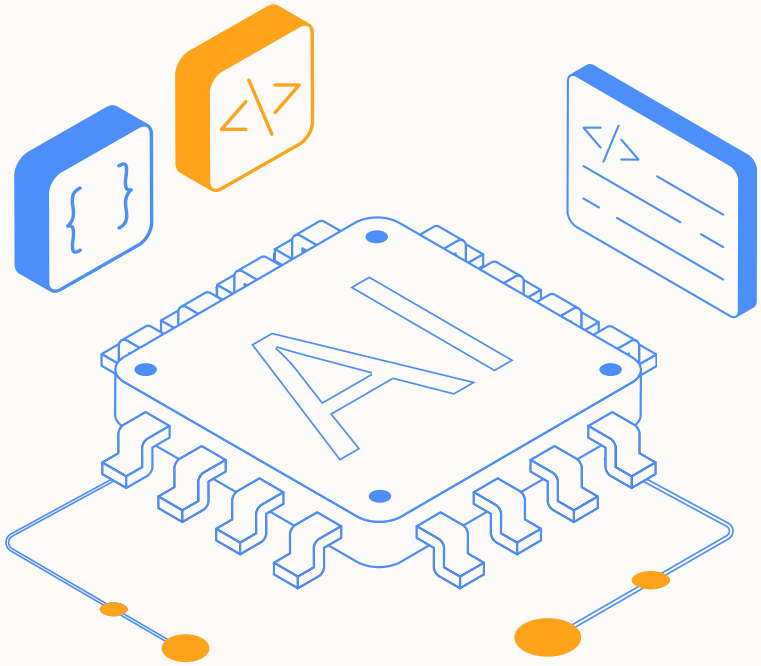
THE STATE OF

Engineering Excellence 2026

# AI Productivity Is Up. The Way We Measure It Is Falling Behind.




harness

harness

# Introduction

Over the past few decades, the technology industry has lived through waves of transformation that redefined how software gets built. The internet changed how software gets distributed.The cloud changed where your code ran. DevOps changed how teams shipped. Each wave was significant. But none of them changed what a developer actually does when they sit down to work.

The emergence of generative AI has changed things at a fundamentally different scale. For the first time, the transformation is happening at the cognitive layer. The moment-to-moment act of writing and deploying software itself has changed.

## *700* Developers and engineering leaders *surveyed*

To understand how organizations are responding, we commissioned a survey of 700 engineering practitioners and their managers from large enterprises across the U.S., U.K., France, Germany, and India. The reality today is that developers are no longer the primary authors of code. They are now the validators of machine-generated output. And beyond validation, the role has expanded in ways that prior tooling never required — deeper scrutiny of code quality and security, greater accountability for downstream outcomes, and more complex judgment calls about when to trust AI and when to override it. The scope of what it means to be a developer has grown, but the productivity frameworks built to measure that work have not kept pace.

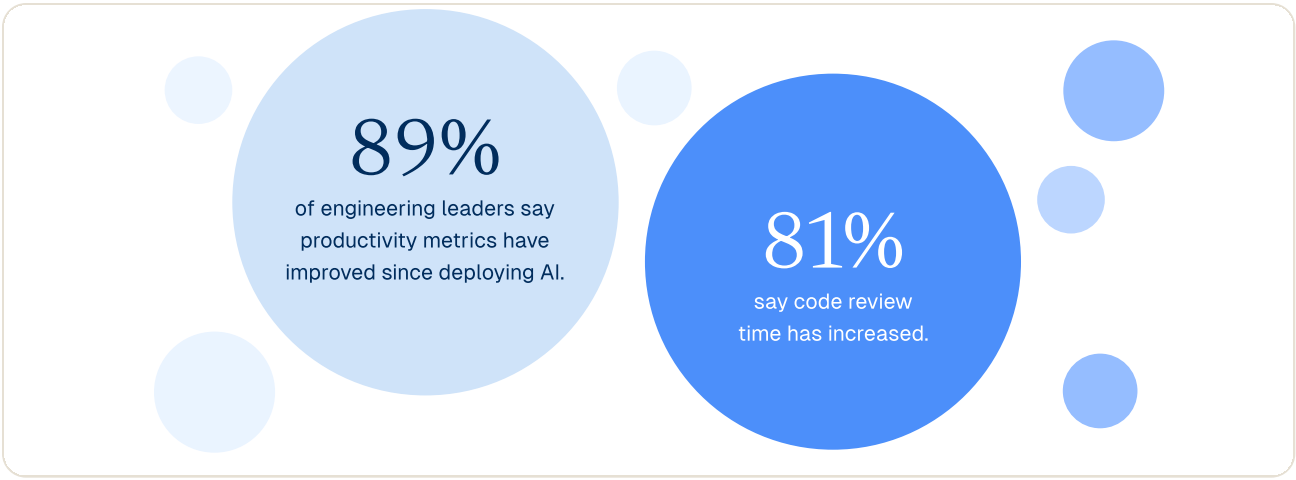
## The *limits* of legacy metrics

Engineering organizations rely on a well-established stack, such as business outcomes, DORA metrics, cycle time, developer experience surveys, and velocity. These frameworks remain foundational. They capture throughput, flow, and team health, and they've earned their place in modern engineering. What they weren't originally designed to measure is the new work AI has introduced, such as validation time, AI agent accuracy, cognitive load, and the trust calibration developers now perform throughout the day.

The opportunity is to extend these frameworks to capture what AI has added to the work, not to replace what is already working.

# The core *disconnect*

Organizations are realizing that measuring AI-era engineering isn't just about output volume, it's about understanding the shift in effort. Unfortunately, most organizations are applying stable-era frameworks to a radically changed environment.

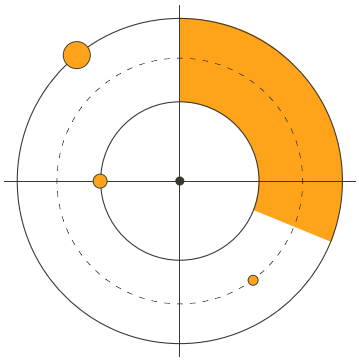



This is the core tension. When AI generates code, output metrics improve and cycle times shorten. Developers report feeling more productive, and in some ways, they are. They're writing more code, solving more complex problems, moving faster through familiar work. But productivity gains don't exist in isolation. They have to show up somewhere.

What the data reveals is that one of the most consistent indicators of where developer time actually goes, code review, is rising sharply at the same time. Code review is toil. It is not the work organizations are trying to accelerate; it is the overhead attached to the work. When 81% of engineering leaders report that code review time has increased since deploying AI, that's not a footnote to the productivity story. It's a direct offset to it. Organizations are measuring gross output. They are not measuring where the productivity gains are actually being spent.

# The *invisible* tax

When validation efforts exist outside the measurement process, they become an afterthought rather than a design principle. This leads to systemic friction that compounds with each deployment.



## 31%

of a developer's day is now consumed by AI-related invisible work that appears in no metric.

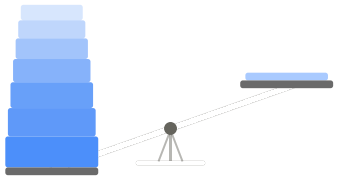
Developers are taking on an unprecedented validation burden. When we asked them where AI creates the most friction, the gaps in current tooling became glaringly obvious:

### Top Sources of AI Friction


Ironically enough, the largest source of invisible overhead — reviewing AI-generated code — is only tracked by 38% of organizations.

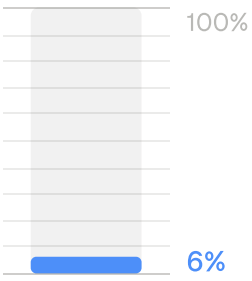
# The AI measurement confidence *paradox*

Without visibility into the true impact of their tools, engineering leaders default to trusting the metrics they have, even when they know those metrics are flawed. When we asked what was missing, the answer was consistent.



**94%**

say tech debt, validation time, and developer burnout are missing from their current metrics.




And only

**6%**

believe their current frameworks are sufficient to close that gap.

Though this is cause for concern, understanding the rationale is critical. When there is no established standard for measuring AI productivity, teams trust the metrics they know simply because they are familiar. High confidence in an incomplete system isn't a signal of accuracy; it’s a signal of misalignment.

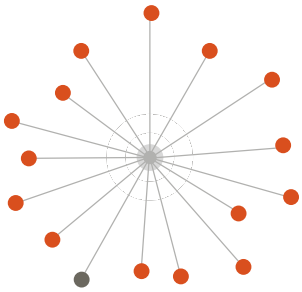
# The AI productivity perception *gap*

Managers and practitioners are using the same AI tools in the same organizations, yet arriving at completely different conclusions. On every measured dimension, leaders report more favorable conditions than the practitioners actually doing the work.

When we asked whether respondents had any concerns about how AI productivity data might be used to evaluate them, the gap was significant.

**15%**

of Managers have no concerns.



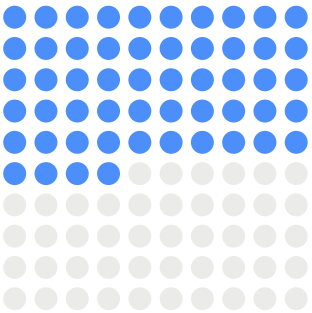
**4%**

of Practitioners have no concerns.

Measurement systems are being designed by the people who feel the safest about them. When frameworks reflect only the leadership view, they systematically undercount operational reality and the pressures developers are actually experiencing.

# Developer trust as the *foundation*

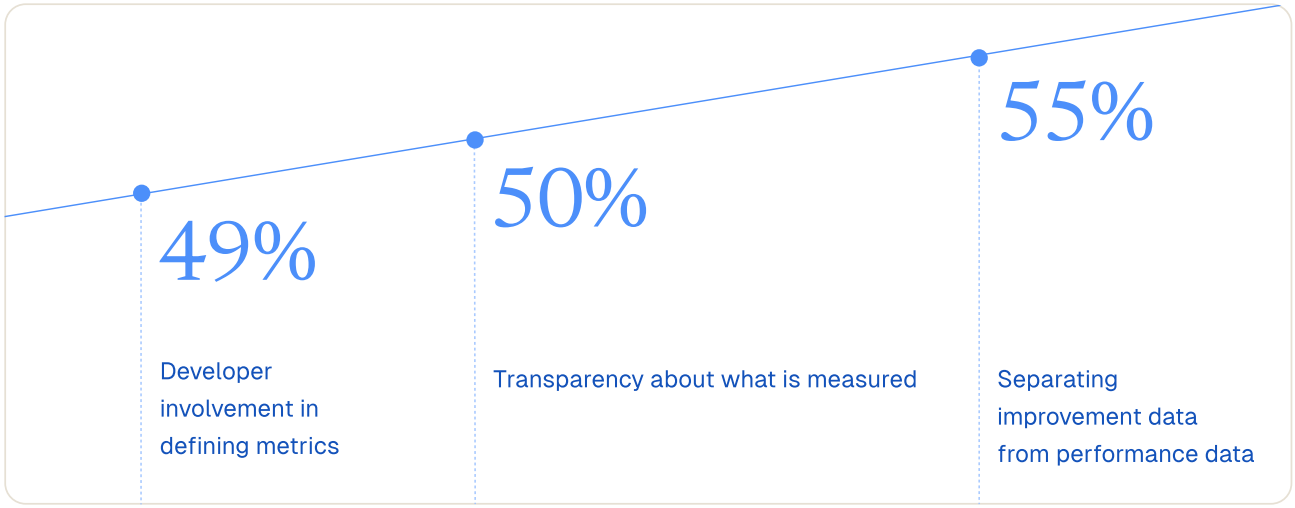
The truth is, systems for measurement are social systems as much as technical ones. Data collected in conditions of distrust does not accurately reflect behavior, it reflects the effort to appear compliant.



**54%**

fear individual performance evaluations based on AI data.

Rather than viewing measurement as a threat, developers actually want to partner with leadership to build better systems. But they need the right guardrails. To feel comfortable with measurement, developers prioritize:




Building trust requires organizational commitment: clear policies about how data will be used, and firm boundaries around how it will not.

# Your next steps *forward*

With AI tooling representing an ever-larger share of engineering budgets, organizations can no longer afford to measure productivity the same way they did before AI entered the workflow.

Integrating holistic, net-effort metrics directly into the software development lifecycle is critical for 2026 and beyond. That said, most organizations aren't starting from a blank slate. Existing systems of measurement are deeply embedded in performance reviews, tooling contracts, and reporting cadences that won't change overnight. The goal isn't to tear down what's working. It's to instrument the gaps alongside it.

Here is where to start:

01 **Measure validation work alongside output:** Track AI review time, debugging overhead, and context-switching costs. A reported 20% gain alongside an unmeasured 31% overhead is a number worth interrogating before it justifies the next investment cycle.

02 **Know your ship rate:** Generating more code is not the same as shipping more value. Organizations need clear visibility into what is actually being completed, merged, and deployed. AI increases code volume. It does not automatically increase delivery.

03 **Treat measurement confidence as a risk signal:** High confidence in an incomplete system should trigger scrutiny, not reassurance. Audit what your frameworks capture versus what AI adoption actually creates.

04 **Design for increasing complexity:** Mature organizations report stronger gains, but also persistent overhead. Plan for more governance, more security reviews, and more measurement sophistication as your AI deployment scales.

05 **Build trust into your system of measurement from the start:** Developers are not resistant to measurement. They are responsive to how it is designed. Build the policy guardrails first, then instrument.