CodeTogether — ENGINEERING INTELLIGENCE

The Management Crisis of AI-Gen Software

How AI Is Breaking Traditional Software Development Controls

Introduction

Management teams are contending with the fallout from roughshod adoption of AI in their software development life cycle (SDLC) – in fact, agentic AI is blowing their SDLC up. While AI has permanently changed software development, its benefits come with high costs, spanning financial, operational, managerial and governance considerations. We take a holistic view of how these changes are playing out and why they are making the black box of software development ever more opaque. The implication is that software organizations need to strategically evolve their ways of planning, tracking and managing.

KEY TAKEAWAYS

- **AI is blowing up the assumptions that modern software development life cycles were built on.**
- **Engineering has shifted from a headcount cost model to a consumption model driven by tokens, agents and compute.**
- **AI is neither good nor bad. It amplifies the capabilities, habits and weaknesses already present in your engineering organization.**
- Organizations can see more code being produced, but fewer can explain whether it is creating business value.
- AI-generated software is arriving faster than teams can review, govern or fully understand it.
- Shadow AI, technical debt and governance gaps are growing alongside productivity gains.
- Most companies are scaling AI adoption faster than they are scaling organizational learning.
- The management challenge is no longer software development alone. It is understanding how humans, AI and software interact across the entire SDLC.

The winners of the AI era will not be the organizations that adopt AI fastest. They will be the organizations that adapt their SDLC fastest.

The Management Crisis of AI-Gen Software

Artificial intelligence is changing software development faster than most organizations can adapt their management systems around it.

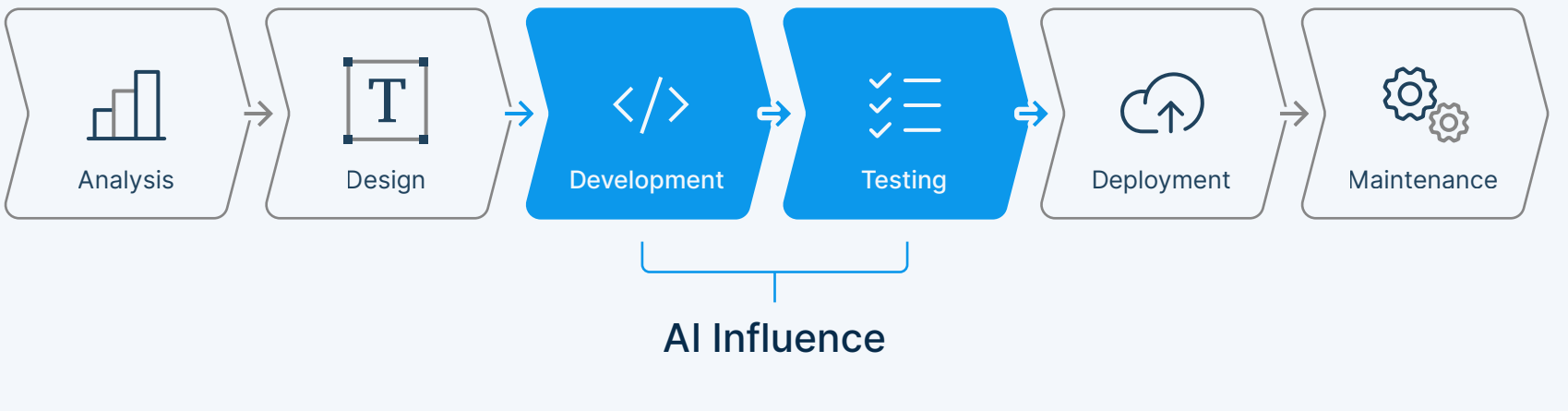
Engineering leaders see enormous opportunity in faster delivery cycles, lower costs and increased developer productivity.

But AI is not simply a new development tool. It is an accelerant. It magnifies whatever already exists inside an engineering organization.

Teams with strong architecture, disciplined engineering practices and effective management often see meaningful gains. Teams operating with weak governance, fragmented systems or limited visibility often find those same problems accelerating alongside the benefits.

The software development lifecycle, or SDLC, was built around human-scale workflows. Review systems assumed humans could inspect incoming code. Governance models assumed software moved at a pace organizations could monitor and control.

AI changes those longstanding assumptions.



AI is disrupting the SDLC, which is configured for human-speed development

Today, developers rely on copilots, chat-based coding tools and autonomous agents to generate code and automate implementation tasks. This changes both the scale and nature of software creation.

Code may now originate from a mix of human decisions, AI suggestions and autonomous agent activity. Output is faster, but often less understood. Traditional review systems, governance models and engineering metrics struggle to keep pace.

As enterprises embrace AI, many still lack visibility into how these tools are being used, what they are costing and where new risks are emerging. The result is an SDLC that still appears familiar on the surface, but increasingly operates under very different conditions underneath.

This paper explores how AI is reshaping the SDLC and why many existing management models are no longer sufficient for AI-assisted development.

01

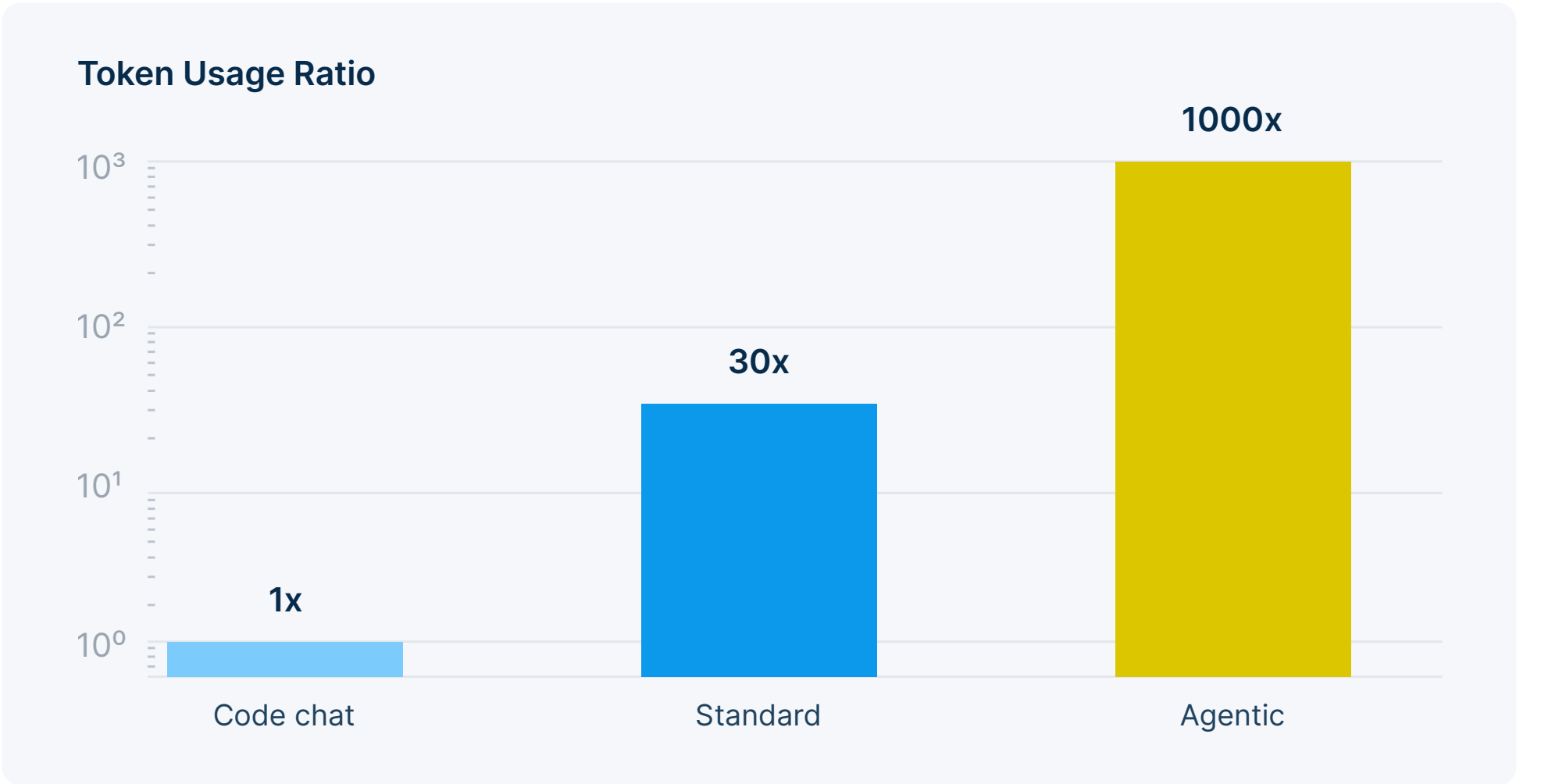
Why is AI making software development costs harder to understand?

The shift from fixed software licenses to variable AI consumption is creating new financial blind spots for engineering leaders.

For decades, software engineering operated on a mostly predictable cost model. Companies hired developers, licensed tools and budgeted around headcount. AI-assisted development has changed that equation almost overnight.

Engineering is now becoming a variable-consumption business driven by tokens, agent activity and usage-based compute costs. That shift introduces a new kind of financial exposure. As AI adoption grows, technology leaders increasingly need to think like CFOs, understanding not just what AI costs, but whether those costs are producing measurable returns. Leaders can see invoices growing, but few can explain which AI activities produced measurable business value and which simply burned resources.

A single poorly structured agent loop can consume thousands of dollars in tokens without producing useful output. One study found that agentic coding workflows can consume up to 1,000 times more tokens than standard code chat interactions, while similar tasks may vary by as much as 30 times in total token usage.[1]



Token usage is highly stochastic and can surge by 1,000x with agentic workflows

The problem expands beyond the outright expense. AI-generated work can also create hidden downstream costs through technical debt, rework and security exposure. In our work with clients, we see organizations frequently “hoping and praying” that AI acceleration is creating net-positive outcomes because they lack visibility into how these tools are being used or whether teams are steering them effectively.

Predictability becomes critical in this environment. Organizations need to test and monitor token consumption as part of their development lifecycle. Without visibility into usage patterns and cost drivers, leaders cannot accurately forecast expenses or determine whether AI-generated output is delivering sufficient ROI to justify ongoing consumption.

In practice, organizations are discovering that AI is actually amplifying management complexity, and often amplifying the true costs of delivery and maintenance.

02

The human factor: AI amplifies the strengths and weaknesses already present.

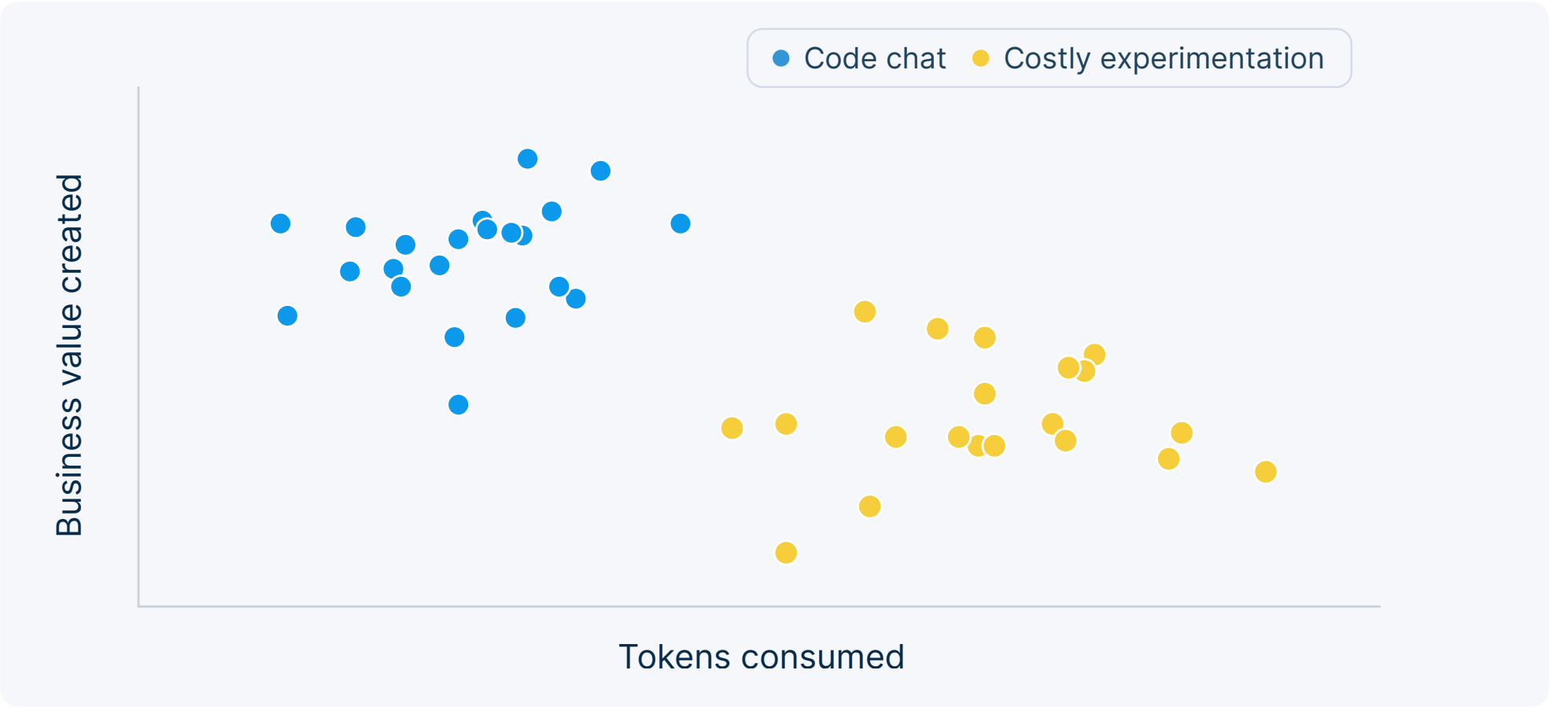
Much of the discussion around AI-assisted development focuses on the technology itself. But in practice, the outcomes often depend less on the AI and more on the people using it. AI is neither inherently beneficial nor harmful. It acts as an amplifier.

In the hands of a skilled engineer with strong architectural judgment, AI can dramatically accelerate development. Repetitive tasks can be completed faster. Complex problems can be explored more efficiently. Experienced developers often use AI to increase productivity while maintaining quality standards and architectural discipline.

But the same technology can produce very different outcomes when applied without sufficient expertise, oversight or context.

One emerging challenge is waste. As organizations adopt token-based AI tools at scale, they are discovering that not all AI consumption creates value. A well-directed engineer may generate significant business impact from extensive AI usage, while another may consume large volumes of tokens pursuing low-value work, inefficient prompts or unnecessary experimentation. In some cases, developers have burned through days of token allocations in minutes while attempting tasks that could have been solved more efficiently through traditional approaches.

AI does not replace engineering culture, management discipline or technical expertise; it accelerates the impact of whatever already exists.



AI usage produces highly uneven outcomes across developers

The challenge is not simply controlling costs. It is understanding which usage patterns create measurable value and which merely consume resources. Without visibility into how AI is being used, organizations cannot distinguish productive adoption from expensive experimentation.

The second challenge is quality.

AI makes it easier than ever for developers to generate large amounts of code, including in systems they may not fully understand. While this can accelerate delivery, it can also be a recipe for technical debt. We frequently hear engineering leaders describe situations where AI-generated changes are layered onto already complex systems, creating what one technology leader called "glue on top of glue on top of glue." The code may function in the short term, but the long-term maintenance burden continues to grow.

This dynamic can be especially problematic in legacy environments. Developers who lack deep knowledge of an application or architecture can now generate substantial modifications quickly, often without fully understanding the downstream implications. The result may be increased rework, growing architectural complexity and additional burden on senior engineers responsible for maintaining system integrity.

In the frenzy to better understand AI's effect on the SDLC, organizations cannot skip over this human element. The more they actively try to measure these dynamics, the better they will understand the interaction between their human workforce, their software and the tools they use to produce and manage it.

03

The new black box: AI is making software development harder to see.

As AI-generated work increases, organizations lose visibility into how software is being created, reviewed and delivered.

Software development has always involved some level of opacity, with leadership relying on inferred measures of production and efficiency.

But most teams managed their workflow despite this opacity, as development operated within human constraints. A developer wrote the code, understood the logic behind it and could usually explain why specific decisions were made. Today, AI-assisted development changes that dynamic dramatically.

Now, engineering leaders can see that AI-driven code was produced, but not why the AI chose a particular approach, what assumptions it made or whether it accounted for the realities of existing systems and architecture.

We frequently hear from technology leaders that this feels like managing an invisible workforce. AI agents can complete meaningful development tasks, but organizations are losing visibility into how work is being created and validated. That uncertainty becomes especially risky when AI-generated code interacts with legacy systems, customer data or security-sensitive environments.

Shadow AI is evading oversight and amplifying software sprawl

This operational opacity is magnified by the rise of “shadow AI.” The 2026 Annual SaaS Benchmark from Torii found that the average enterprise uses 830+ applications, with more than 60% operating outside of IT oversight. Nearly all of the top-ranked “shadow” applications are pureplay AI tools.[2] In many organizations, shadow AI emerges when pressure to adopt AI outpaces the governance structures needed to evaluate, standardize and oversee how these tools are used.

Leading organizations are responding by creating formal AI governance and solution architecture functions that evaluate AI tools for cost, integration, risk and business value before they become embedded in day-to-day workflows.

In engineering teams, this operational opacity is amplifying the disconnect between what engineering leaders think is happening inside their software development lifecycle and what is actually occurring beneath the surface.

04

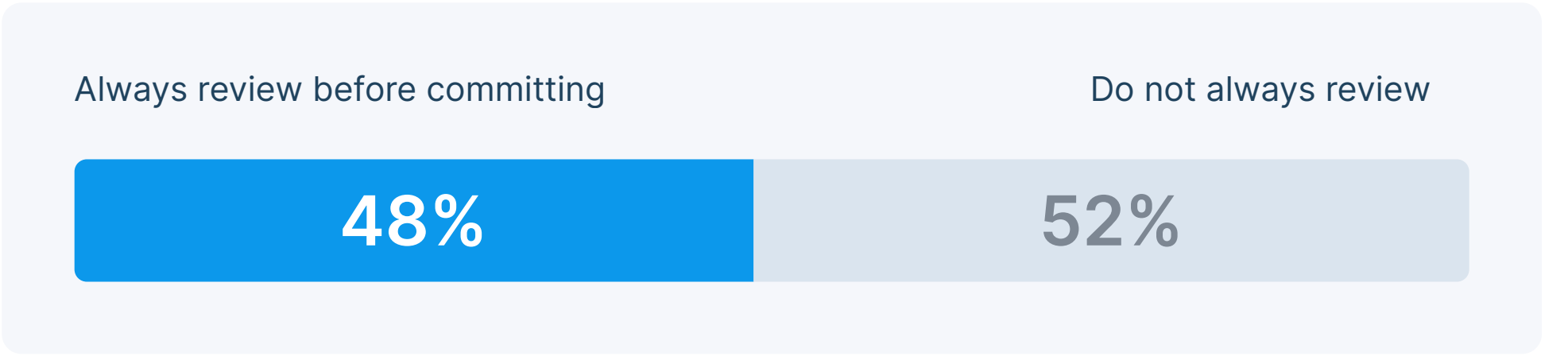
Why can’t human review processes keep up with AI-generated code?

AI can produce software faster (and in larger batches) than humans can validate it, creating new quality and oversight risks.

Modern AI coding tools can generate software far faster than humans can realistically review it. That imbalance is creating a growing bottleneck across the software development lifecycle.

Large pull requests have always been difficult to review thoroughly. But AI-assisted development dramatically increases both the volume and size of code submissions. Developers can now generate thousands of lines of code in minutes, often bundling larger and more complex changes into a single review cycle. As throughput accelerates, human reviewers struggle to keep pace.

The result is what many engineering leaders are experiencing already: review collapse. Just 48% of developers say they always check AI code before committing, according to a 2026 survey by Sonar.[3] When code does go through review, developers often say it is harder to catch errors because the code typically “looks right” – and in fact it takes more time to review than human-generated code.



Less than half of developers check AI code before committing

When review queues become overloaded, teams naturally begin to shortcut the process. In many cases, code is effectively rubber-stamped because the alternative would slow delivery to an unacceptable pace.

Many organizations initially assume the solution is simply adding more reviewers. But review collapse is not a staffing problem. It is a systems problem. If AI accelerates code generation upstream without visibility, governance and process controls throughout the lifecycle, organizations simply push larger volumes of unvalidated work into an already strained review pipeline.

In practice, the organizations adapting most successfully are not trying to review more code manually. They are building better visibility into how code is created, how AI is being used and where risk is accumulating before review ever begins. In this model, the role of the developer shifts from primary reviewer to auditor. Humans still provide oversight, spot-check outputs and validate critical decisions. As AI accelerates software production, scalable oversight depends on combining automated review with targeted human judgment.

05

AI governance is lagging behind AI adoption.

Most organizations are deploying AI-assisted development faster than they can establish the controls needed to manage risk.

Most organizations did not adopt AI through a carefully governed transformation plan. They adopted it because teams were already using it and/or there was competitive pressure to drive AI adoption.

With this backdrop, governance frameworks, reporting systems and oversight processes are barely established in most organizations, if at all.

That creates a growing maturity gap, and it matters to boards and investors. Suddenly, leaders must answer difficult questions they were never previously required to quantify:

- Are AI tools reducing costs or increasing them?
- Are they improving code quality or introducing hidden technical debt?
- Are teams following consistent practices, or improvising independently across the organization?

The challenge becomes even more urgent in regulated industries such as finance, health care and defense, where traceability, accountability and human oversight are critical. As AI-generated work becomes more deeply embedded into enterprise systems, organizations increasingly need defensible ways to explain how software was created, reviewed and approved. In a 2026 Deloitte report, industry surveys show that 74% of companies plan to deploy agentic AI in the near term, but only 21% have a mature model to govern such tools.[4]

Today, many companies are still operating on trust and assumption rather than measurable visibility. Leaders may know AI is being used widely across engineering teams, but they often lack the telemetry needed to evaluate risk, enforce standards or confidently report outcomes to boards and stakeholders.

Still, it's crucial that AI governance be agile and speed-enabling, not bureaucratic. The organizations adapting most successfully are treating governance not as a compliance exercise, but as an operational capability that evolves alongside AI adoption. Effective governance is not designed to slow innovation. It is designed to help organizations move faster with confidence by providing the visibility, accountability and decision-making frameworks needed to manage AI at scale.

This increasingly requires close partnership between technology and AI leadership, with governance frameworks that can adapt to changing tools, risks and business priorities rather than relying on static policies that quickly become obsolete.

In the AI era, governance can no longer rely on inference alone. New telemetry is urgently needed to track, manage and plan around AI activity.

06

Are engineering organizations ready for AI-driven development?

Many leaders are pursuing AI productivity gains without redesigning the workflows, processes and management systems required to support them.

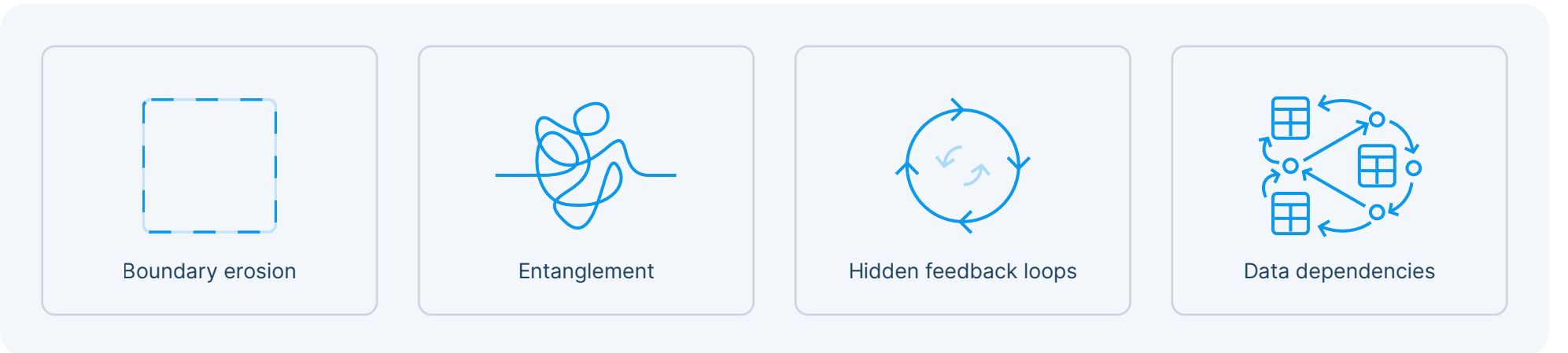
Many organizations entered the AI era assuming the primary challenge would be adopting new tools. In reality, the bigger challenge is whether existing engineering environments are prepared to support AI-driven development effectively.

Could AI deliver the productivity gains and cost reductions that it promised? Sure, but those outcomes depend heavily on the maturity of the SDLC environment. AI does not replace or even improve underlying engineering conditions. It amplifies them.

Teams with strong documentation, clear architecture, disciplined review practices and well-structured repositories often see meaningful acceleration from AI-assisted development. Teams operating in fragmented or poorly governed environments may see the opposite: higher token consumption, inconsistent outputs, growing technical debt and increased operational risk.

The same principle applies to data. AI systems are only as effective as the data, architecture and governance frameworks that support them. Organizations that have not established strong foundations in these areas often discover that AI amplifies existing data quality issues, process gaps and architectural weaknesses rather than solving them.

Researchers have found that the integration of AI into coding is introducing whole new categories of technical debt – long-term maintenance challenges and hidden costs arising from things like boundary erosion, entanglement, hidden feedback loops and data dependencies.[5]



AI coding introduces new technical debt categories

The danger is that many companies are already moving quickly without understanding these dependencies. The pressure to accelerate development, reduce costs and “add AI” into workflows is immediate. But without visibility into how AI is affecting the full software development lifecycle, organizations may unintentionally scale inefficiency, technical debt and security exposure alongside productivity gains.

The organizations seeing the strongest long-term results are not slowing AI adoption. They are first conducting an honest assessment of their data, architecture and development environments, then building the operational visibility needed to guide AI adoption responsibly and at scale.

07

Why most organizations cannot measure what AI is changing

Traditional engineering metrics reveal activity, but they often fail to show whether AI is improving outcomes or creating new forms of waste.

Traditional engineering metrics reveal activity, but they often fail to show whether AI is improving outcomes or creating new forms of waste.

Today, few leaders can see their AI usage with a broad enough view to understand how it is helping and how it is hurting their core software development.

Engineering leaders can often see activity increasing. More code is being generated. More pull requests are moving through the pipeline. Teams appear faster and more productive. But most organizations lack the systems needed to measure whether those changes are actually improving outcomes.

This creates a dangerous learning gap. Without visibility into how AI is being used across the software development lifecycle, companies struggle to answer basic operational questions.

- Which workflows are creating measurable productivity gains?
- Which are increasing review burden or technical debt?
- Which teams are using AI effectively, and which are generating expensive rework?
- How do we train or direct our developer team to double down on what's working and fix what's not?

In many organizations, those answers remain largely unknowable. They simply do not have the telemetry to measure what matters in the new AI era.

We frequently hear leaders describe AI adoption as "running downhill trying to keep up." Teams are deploying new tools rapidly because competitive pressure feels urgent, but few organizations have established a defensible framework for understanding what is actually happening inside their engineering systems.

The result is that many enterprises are scaling AI usage without simultaneously scaling organizational learning. Over time, that makes it increasingly difficult to separate genuine productivity gains from hidden operational risk.

The most effective organizations approach AI adoption as a continuous learning process. They establish baseline measurements, compare workflows with and without AI assistance, and monitor not only development activity but also downstream impacts on cost, system performance, capacity and user experience. They expect some experiments to fail and use telemetry to identify issues early, adjust course and improve outcomes over time.

In the absence of evidence-based measurement, organizations risk making AI decisions based on assumptions rather than demonstrated business value and ROI.

CONCLUSION

Leaders in the AI era are prioritizing adaptation of their SDLC

Today, a single developer using AI agents can generate the output that once required entire teams. Code can be produced faster than organizations can review, govern or fully understand it. Yet many companies are still operating with workflows, metrics and oversight models built for a pre-AI era.

This is why so many organizations feel growing friction inside their engineering systems. The challenge is not in the adoption of AI tools. It is in the adaptation of the entire SDLC to a new operating environment where visibility, orchestration and governance matter as much as raw development speed.

The organizations that succeed in this transition will not necessarily be the ones generating the most code. They will be the ones best able to understand, measure and guide how AI changes work across the full engineering lifecycle.

The winners adapt their SDLC fastest — they don’t just adopt AI fastest.

What winning looks like in the AI era

The organizations navigating AI-assisted development most effectively share a common trait. They did not simply adopt AI tools. They rebuilt the visibility, governance and measurement systems around them.

That means treating AI adoption as an operational transformation, not a tooling decision. It means establishing baseline measurements before scaling. It means building telemetry that shows not just how much code is being produced, but whether that code is creating business value, accumulating technical debt or consuming resources without proportional return.

It also means accepting that AI will keep moving faster than governance naturally wants to. The goal is not to slow adoption. It is to build the organizational capacity to learn, adapt and course-correct at the same pace AI is changing the work.

The companies that win in this environment will not be the fastest to deploy AI. They will be the fastest to understand what AI is actually doing inside their engineering organization, and to act on that understanding with discipline.

What to do next

If the dynamics described in this paper feel familiar, the most valuable first step is establishing a clear baseline. Before optimizing AI adoption, organizations need honest answers to a few foundational questions:

- How much of our current AI usage is producing measurable business value, and how much is generating cost without clear return?
- Where are review and governance processes already under strain, and how will AI scale that strain?
- Do we have the visibility to answer board-level questions about AI cost, quality and risk with confidence?

Organizations that can answer those questions clearly are positioned to scale AI adoption responsibly. Those that cannot are scaling risk alongside productivity.

About CodeTogether

CodeTogether gives engineering leaders accurate, independent data about how their teams are performing and whether their AI tools are producing real results. Its platform captures developer activity across coding environments, tools and workflows, measuring active coding time, AI code usage, team health and delivery risk in real time. Its AI-specific layer, AITrax, tracks token spend, AI code retention and developer adoption across every AI tool in the organization, so technology leaders can answer the question every board is asking: is AI actually working for us?

Book a demo at **codetogether.com**.

Source: arxiv.org/abs/2604.22750

Source: toriihq.com/saas-benchmark-annual-report-2026

Source: sonarsource.com/state-of-code-developer-survey-report.pdf

Source: deloitte.com/us/en/what-we-do/capabilities/applied-artificial-intelligence/content/state-of-ai-in-the-enterprise.html

Source: arxiv.org/abs/2405.11825