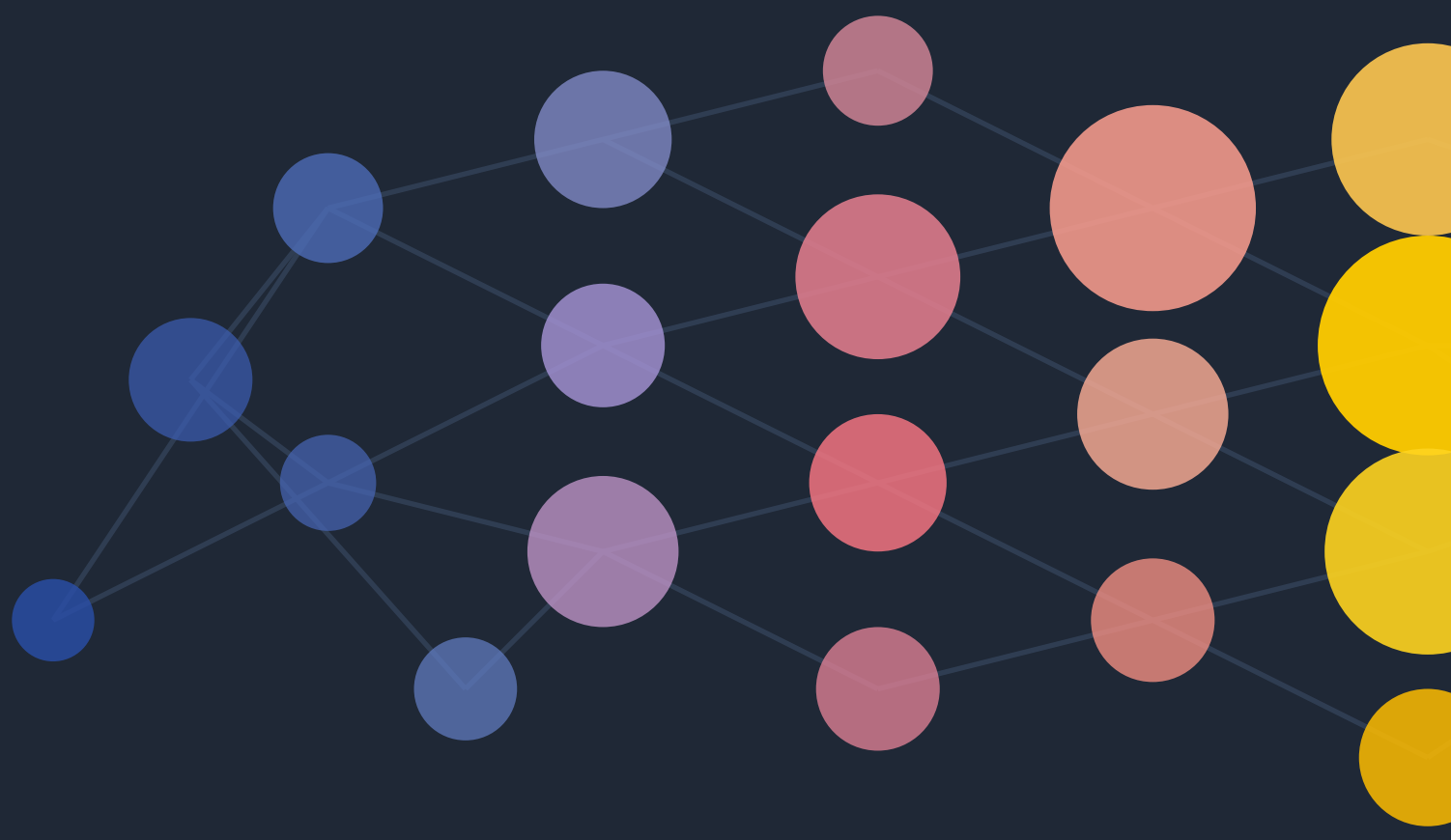
Quotient

An AI Maturity Model for Engineering Teams

A research-driven framework for understanding how engineering organizations adopt AI



2026 · getquotient.com

Table of Contents

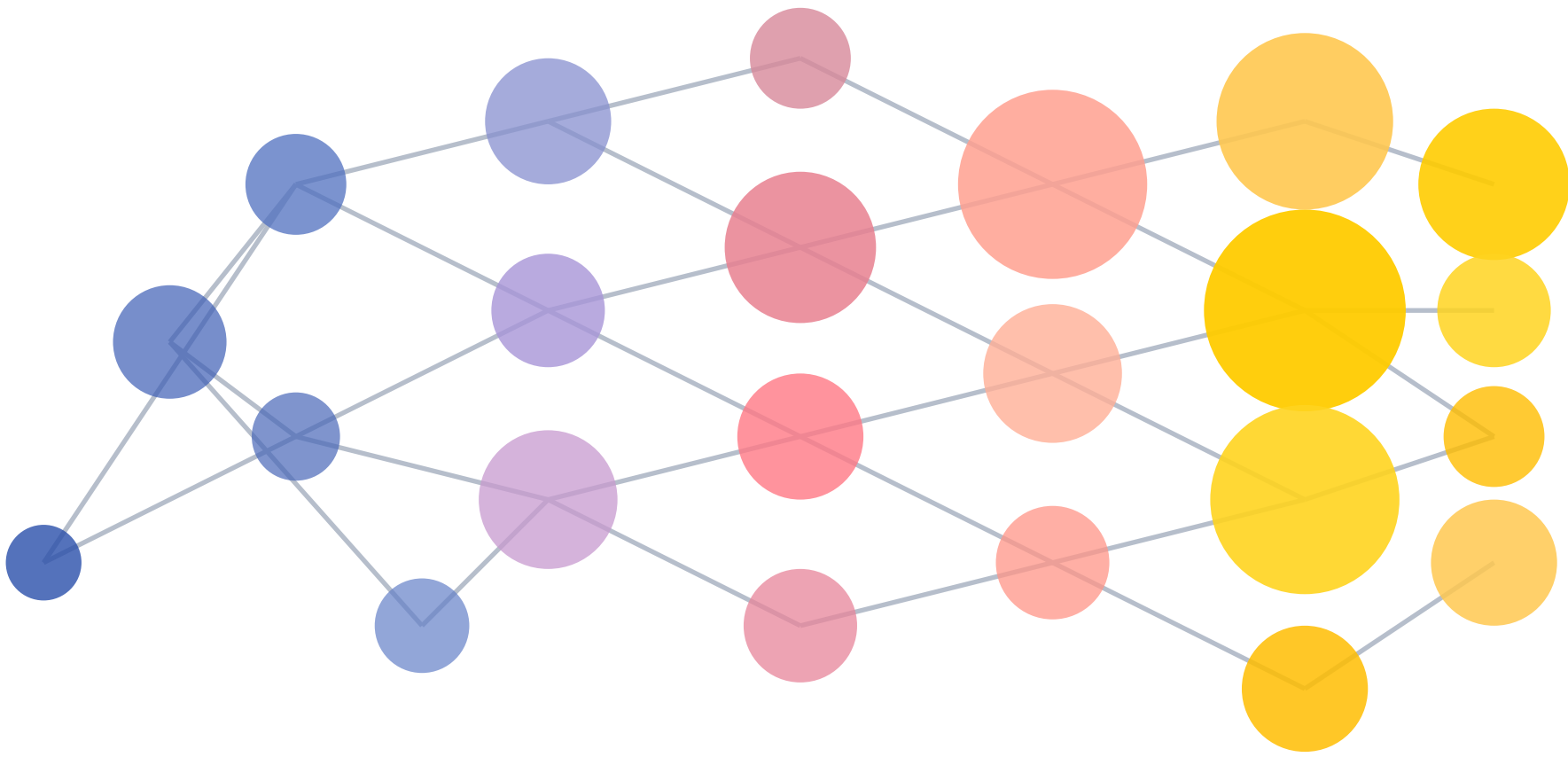
Executive Summary

Artificial intelligence is rapidly changing how software is built. AI capabilities now extend beyond code generation into testing, documentation, debugging, and planning, reshaping workflows across the software development lifecycle. As adoption accelerates, many engineering leaders are asking two critical questions: How mature is our AI adoption today, and is it actually improving engineering outcomes?

Some teams report meaningful productivity gains, while others see little improvement or even new sources of instability and risk. Research suggests the difference often lies not in the tools themselves, but in how organizations integrate them into their engineering systems. Teams that see the greatest benefit from AI build supporting capabilities around governance, validation, workflow integration, and access to internal context.

This whitepaper introduces the **AI Adoption Maturity Model for Engineering Organizations**, a research-driven framework describing how organizations adopt AI across five stages of capability and operational risk. The model synthesizes insights from dozens of studies on developer productivity and AI-assisted development, informed by collaboration with researchers and analysis of academic studies summarized in the Research-Driven Engineering Leadership series.

In the sections that follow, we outline the five stages of AI adoption, the capabilities that define each stage, and how engineering leaders can assess their organization's maturity and determine the next improvements required to increase AI's impact while managing operational risk.



Introducing the AI Adoption Maturity Model

The AI Adoption Maturity Model describes how engineering organizations evolve as AI becomes embedded across the software development lifecycle. It evaluates six organizational capabilities that shape how AI is integrated into engineering systems, defining five stages of maturity ranging from ad hoc experimentation to increasingly automated development workflows.

These capability areas reflect recurring patterns identified in research on developer productivity and AI-assisted software development. Across numerous studies, successful AI adoption consistently depends on how organizations enable developers, govern AI usage, validate outputs, integrate AI into workflows, automate tasks safely, and provide AI systems with access to meaningful internal context.

STAGE ATTRIBUTES	STAGE 1 Ad Hoc Adoption	STAGE 2 Assisted Development	STAGE 3 Standardized Workflows	STAGE 4 Supervised Automation	STAGE 5 End to End Autonomy
Stage Profile	Individuals experiment with AI tools on their own initiative, using personal accounts. The organization has limited or no formal policies, training, or infrastructure to support AI adoption.	Engineers use AI collaboratively but adoption remains inconsistent. The organization enables AI adoption through centralized tools, learning opportunities, and early policies.	AI is embedded into multiple workflows across the SDLC triggered by engineers with guardrails to maintain quality. Teams operate with shared processes and leverage system integrations.	Teams use autonomous agents for bounded tasks that trigger automatically. Agents handle low complexity work, with human oversight for higher-risk decisions.	AI orchestrates end-to-end workflows across engineering systems. Humans focus on oversight, high-leverage decisions, and managing exceptions. The organization operates as AI-native.
Enterprise Adoption Risk Operational risk in enterprise environments	High	Low	Low	Medium	High
CHARACTERISTICS					
Enablement Learning and skill development	• Individual experimentation; no formal training or standards	• Informal learning about AI tools and prompting techniques	• Formal AI training established; shared practices across teams	• AI literacy is widespread; teams confidently experiment and refine usage	• Continuous AI skill development is embedded in engineering culture
Policy and Governance Organizational guidance and guardrails	• No guidance on acceptable AI usage	• Basic or inconsistent AI usage policies across teams	• Clear organizational expectations for how AI is used in development	• AI usage policy is aligned with the company's broader strategy and goals	• The organization continuously updates AI policies as systems evolve
Validation & Testing Ensuring quality of AI-generated work	• Developers manually review AI-generated output	• Basic validation checklists for AI-generated work	• Automated checks validate AI-generated code	• Dedicated AI agents validate AI-generated work before release	• User feedback and system signals continuously improve AI validation and testing
Embedding in Workflows AI integration in developer workflows	• AI used outside engineering tools	• IDE copilots and chat assistants widely used	• AI used consistently across development workflows	• AI agents coordinate across engineering systems	• AI orchestrates workflows across the development lifecycle
Workflow Automation Automation of development workflows	• AI not used to automate engineering tasks	• Developers manually trigger AI to assist tasks	• Systems automatically trigger AI for specific tasks.	• Agents execute bounded engineering tasks automatically	• AI orchestrates multi-step engineering workflows autonomously
Data Context & Access Internal data available to AI systems	• AI tools cannot access internal documentation or codebases	• Developers manually provide internal context to AI tools.	• AI tools can access parts of the codebase or documentation	• AI automatically retrieves relevant internal context	• AI operates on a structured internal knowledge system across tools

Figure 1: The AI Adoption Maturity Model for Engineering Organizations

The Five Stages of AI Maturity

1 Ad Hoc Adoption

Profile: AI adoption is driven by individual experimentation rather than organizational support. Developers explore AI tools independently, often without formal guidance, infrastructure, or training.

Typical behaviors: AI is used outside normal engineering workflows, and developers manually review their outputs. Policies are unclear, validation practices are informal, and AI systems cannot access internal codebases or documentation.

Operational risk: High — unstructured AI use can create security and reliability issues.

2 Assisted Development

Profile: AI tools begin to spread across teams as organizations enable early adoption. Engineers increasingly use AI to assist with coding and documentation, but usage patterns remain inconsistent.

Typical behaviors: Teams adopt AI copilots and begin establishing basic usage policies. Developers still manually trigger AI assistance and provide most of the context needed for AI systems to be effective.

Operational risk: Low — AI primarily assists developers rather than acting autonomously.

3 Standardized Workflows

Profile: AI becomes embedded in development workflows and supported by shared organizational practices. Engineering teams adopt consistent expectations for how AI tools are used.

Typical behaviors: AI assists across multiple stages of development and is integrated into common engineering tools. Automated validation checks begin verifying AI-generated work.

Operational risk: Low — AI usage is standardized and supported by clear guardrails.

4 Supervised Automation

Profile: AI systems begin executing bounded engineering tasks automatically. Autonomous agents handle lower-complexity work while engineers oversee higher-risk decisions and system behavior.

Typical behaviors: AI agents coordinate across engineering systems, retrieve internal context automatically, and execute defined development tasks without manual triggers. Validation systems monitor AI-generated work and escalate issues.

Operational risk: Medium — automation expands and AI systems begin acting autonomously.

5 End-to-End Autonomy

Profile: AI orchestrates multi-step engineering workflows across systems with minimal human intervention. Engineers shift toward supervising systems, making high-level decisions, and managing exceptions.

Typical behaviors: AI systems coordinate complex workflows across the software development lifecycle and operate on structured internal knowledge about codebases, systems, and processes. Validation systems continuously learn from production signals.

Operational risk: High — autonomous systems can create cascading failures if governance is insufficient.

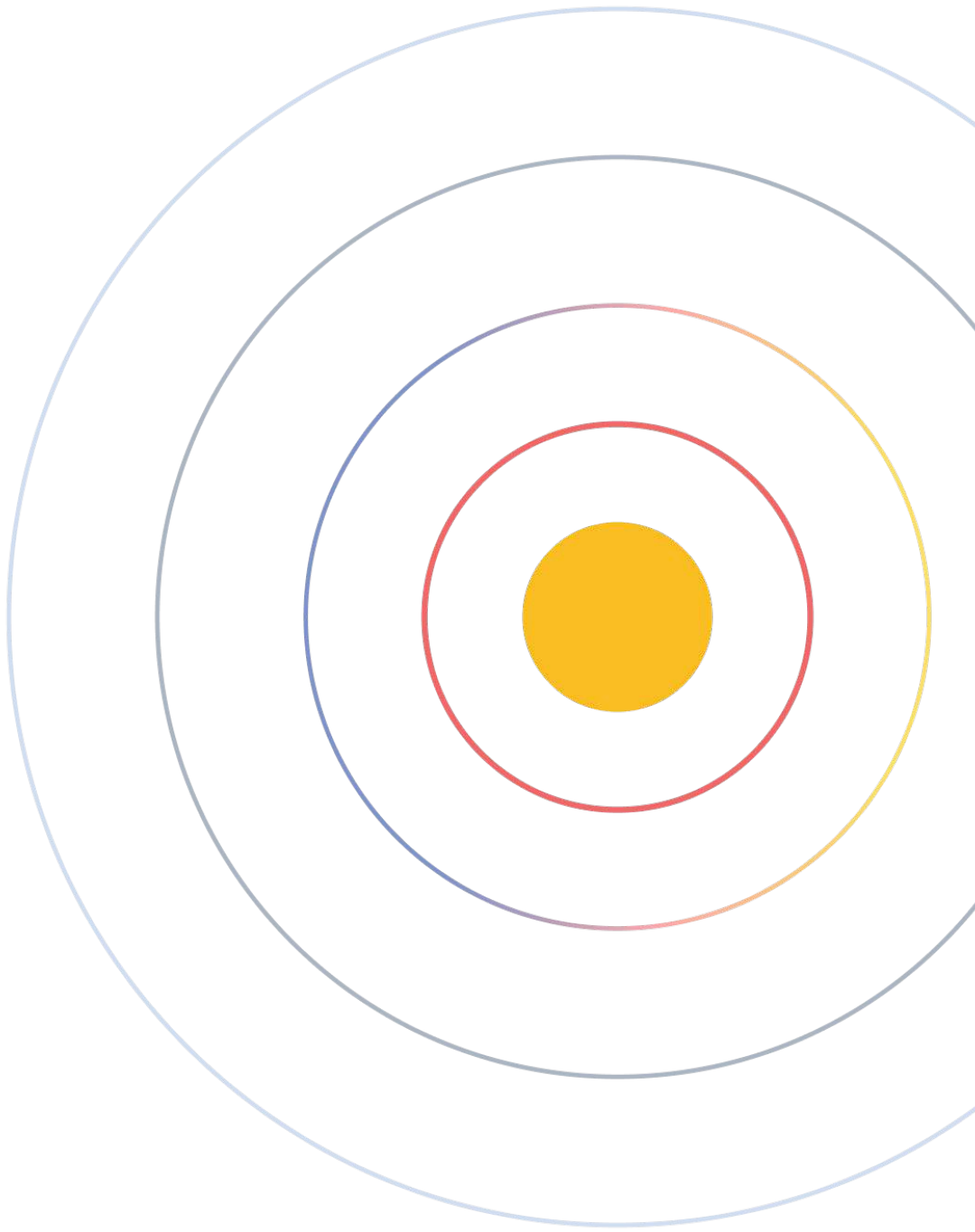
Why the "Most Mature" Stage Isn't Always the Right Goal

Greater AI maturity does not always lead to better outcomes. While advances in AI agents suggest a future of increasingly autonomous development systems, higher levels of automation also introduce new forms of operational risk. As AI takes on more complex tasks, failures can propagate more quickly and become harder to detect, particularly when validation and oversight systems have not evolved at the same pace.

Key Insight

For many engineering organizations, the optimal near-term target may be **Stage 3 or Stage 4**. At these stages, AI is embedded in engineering workflows and can automate well-defined tasks while engineers remain responsible for higher-risk decisions and system oversight. This balance allows teams to capture meaningful productivity gains while maintaining reliability and product quality.

Research on complex software systems shows that reliability depends not only on automation, but on the systems that monitor and validate that automation. As organizations move toward more autonomous workflows, they must invest in stronger validation, governance, and oversight capabilities. Part of the risk at higher stages also stems from the fact that the role of humans in software development is still evolving. As AI systems take on more routine implementation work, engineers will increasingly shift toward higher-level responsibilities such as system design, validation, and decision making.



How Engineering Leaders Can Use the Model

To apply the model in an engineering organization, leaders can use it as a diagnostic tool to evaluate how AI is integrated across their development systems and where to invest next. The first step is to assess where the organization sits across the six capability areas. Because teams often operate at different stages of maturity, leaders should focus on the overall pattern and prioritize the areas that most constrain system-wide performance.

Leaders should also be cautious about relying solely on self-assessment. Research shows that individuals and organizations systematically overestimate their own effectiveness, particularly at senior levels. Rely on external signals and objective metrics to more accurately diagnose maturity.

Once the current stage is identified, leaders can discover the capability gaps preventing progress. Many organizations adopt AI tools quickly but lag in the supporting systems and development workflows that allow teams to see the end-to-end improvement in productivity.

Finally, leaders can prioritize targeted investments that unlock the next stage of maturity and identify objective metrics to track impact of their investments. By measuring how these changes affect overall productivity, organizations can clearly see whether their investments are improving outcomes and where further improvements are needed. (See Page 7 for insights on how to measure AI impact)

STEP 1

Assess Capabilities

Determine where the organization sits across the six capability areas. Focus on averages, not outliers.

STEP 2

Identify Gaps

Identify the capability gaps preventing progress to the next stage of maturity.

STEP 3

Target Improvements

Prioritize targeted investments to unlock the next stage of maturity.

Case Example: Advancing AI Maturity in Practice

An engineering organization adopting AI coding tools initially operated at Stage 2: Assisted Development. Developers regularly used AI to generate code and documentation, but usage patterns varied widely across teams. AI outputs were manually validated, policies were still evolving, and AI tools lacked access to internal documentation and system context.

As a result, while developers reported faster coding and increased individual productivity, teams saw limited improvements in overall delivery performance. Code review queues grew, testing cycles slowed, and the additional verification required for AI-generated work introduced new friction across the development lifecycle.

Using the AI Adoption Maturity Model, leaders identified the capabilities required to move toward Stage 3: Standardized Workflows. They introduced clear expectations for AI usage, integrated validation checks into development workflows, and improved access to internal documentation that AI tools could reference.

These changes embedded AI more consistently across the software development lifecycle and reduced downstream bottlenecks. Over time, the organization saw improvements in delivery throughput, shorter cycle times for code changes, and higher developer satisfaction as AI became a reliable part of everyday workflows.

Measuring the Impact of AI Adoption

Recent research highlights an important pattern in AI-assisted development: AI often improves individual productivity faster than it improves team performance. The 2025 DORA AI Capabilities Report found that developers frequently report higher personal productivity when using AI tools. At the same time, some organizations experience declining team-level outcomes as faster code generation creates new pressure on downstream processes such as code review, testing, and integration.

This creates a measurement challenge for engineering leaders. Many organizations track AI usage metrics, such as the percentage of developers using AI tools or the number of AI-generated suggestions accepted. While useful, these metrics reveal little about whether AI is improving delivery speed, system reliability, developer experience, or collaboration across teams. AI often amplifies the strengths and weaknesses already present in an engineering system.

Connecting AI Adoption to Engineering Outcomes

AI Adoption Signals	Engineering Effectiveness	Questions Answered
→ Tool usage patterns	→ Change failure rate	→ Where is AI improving outcomes?
→ Workflow integration depth	→ Cycle time	→ Where are new bottlenecks?
→ Output quality	→ Code quality	→ Which capabilities unlock progress?
→ Adoption friction	→ Perceived Productivity	

Figure 2: Sample framework for connecting AI signals to engineering outcomes

Turning AI adoption into measurable engineering outcomes requires analyzing signals across the software development lifecycle to identify where productivity improves and where new bottlenecks emerge. Instead of focusing only on usage metrics, organizations must connect AI adoption signals to delivery performance, code quality, developer experience, and system reliability.

Quotient helps organizations operationalize this approach. Quotient uses engineering signals to measure how AI adoption affects engineering outcomes, identify bottlenecks across development workflows, and uncover the drivers of engineering productivity. These insights help leaders determine where their organization sits within the AI Adoption Maturity Model and which capabilities will unlock the next stage of improvement.

Conclusion

AI is rapidly reshaping how software is built, but its impact depends on how effectively organizations integrate it into their engineering systems. The AI Adoption Maturity Model provides engineering leaders with a practical way to understand where their teams stand today, the capabilities required to progress safely, and how to balance automation with operational risk.

By pairing this framework with clear measurement of AI's impact on engineering outcomes, organizations can move beyond experimentation and build the systems, practices, and guardrails needed to unlock the full potential of AI in modern software development.

Learn more about measuring AI's impact on engineering outcomes

getquotient.com/ai-impact

Appendix: Research Citations

The following citations represent a sample of the research analyzed in developing this model. To read summaries of these papers and review additional papers studied, visit **rdel.substack.com**.

1. DORA State of AI Assisted Software Development
   https://dora.dev/research/2025/dora-report/
2. Beyond the Commit: Developer Perspectives on Productivity with AI Coding Assistants
   https://arxiv.org/pdf/2602.03593
3. Understanding Dominant Themes in Reviewing Agentic AI-authored Code
   https://arxiv.org/pdf/2601.19287
4. AI Agents and Higher-Order Work
   https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5713646
5. How much does AI impact development speed? An enterprise-based randomized controlled trial
   https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=11121676
6. The SPACE of AI: Real-World Lessons on AI's Impact on Developers
   https://arxiv.org/pdf/2508.00178
7. AI Where it Matters: Where, Why, and How Developers Want AI Support in Daily Work
   https://arxiv.org/pdf/2510.00762
8. Measuring the Impact of Early-2025 AI on Experienced Open Source Developer Productivity
   https://metr.org/blog/2025-07-10-early-2025-ai-experienced-os-dev-study/
9. Self–other rating agreement in leadership
   https://doi.org/10.1016/j.leaqua.2010.10.006

We thank the researchers who provided feedback on this framework, including Dr. Jenna Butler, Dr. Cat Hicks, and Brian Houck.