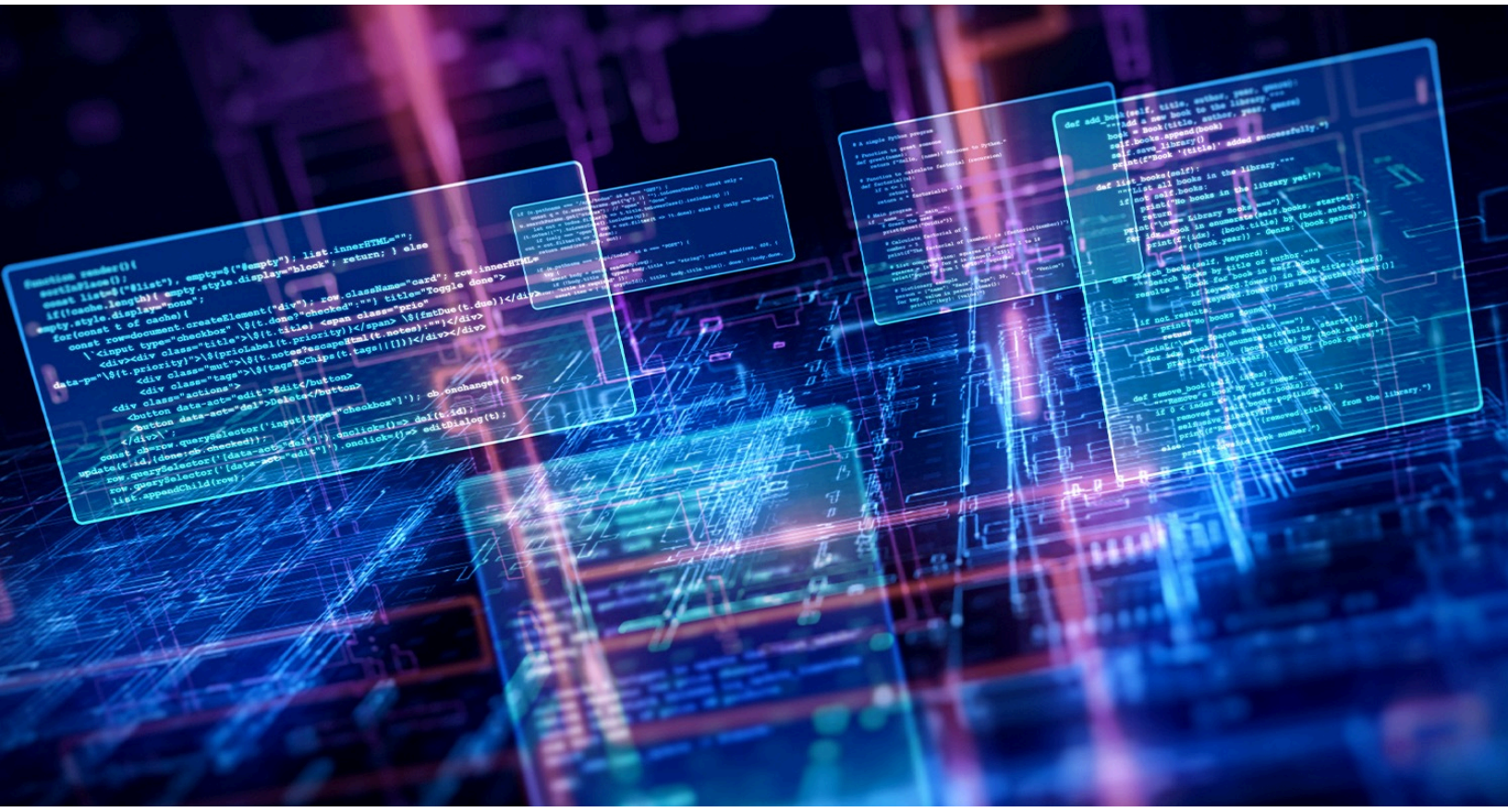
McKinsey & Company

Technology, Media & Telecommunications Practice

Unlocking the value of AI in software development

Many software teams are seeing some impact from AI, but a select group is experiencing more significant gains by rethinking roles, processes, and performance enablers.

This article is a collaborative effort by Charlotte Relyea, Martin Harrysson, and Matt Linderman, with Jose Mario Pena, Nandita Bothra, and Natasha Maniar, representing views from McKinsey's Technology, Media & Telecommunications Practice, McKinsey Technology, and QuantumBlack, AI by McKinsey.



November 2025

For all of software's technological advances and world-changing impacts over the past half century, its seismic potential has historically been limited by a shortage of skilled developers, finite coding capacity, and the complexity of coordinating large projects. The emergence of gen AI, and more recently agentic AI, was and is supposed to overcome those obstacles, leading to untold new productivity and value creation. While many organizations are already seeing some positive impact from these tools, a small subset of companies is reaping particularly large gains. That is one of the key findings from a recent McKinsey survey of a wide range of nearly 300 publicly traded companies.

To help understand what distinguishes the highest-performing AI-driven software organizations, we assessed AI adoption levels, outcomes, and practices among developers and product management professionals (see sidebar, "About the research"). The highest performers saw a notably large impact from AI across four key development metrics: team productivity, customer experience, and time to market (16 to 30 percent improvements), as well as software quality (31 to 45 percent). Our research shows that realizing the revolutionary promise of AI on software product development will take much more than adoption—it will require a complete overhaul of processes, roles, and ways of working to keep pace with accelerating tool and model intelligence. This article examines two key shifts and three essential enablers that high-performing software organizations use to maximize the potential of incorporating AI into software development.

About the research

Our survey methodology covered five core dimensions of the product operating model: structure, strategy and governance, ways of working, culture and talent, and tooling. We asked nearly 300 senior leaders from publicly traded companies about AI adoption and practices, and 100 of them assessed impact and performance across four outcomes: software quality, time to market, team productivity, and customer experience. Top performers were defined as the top quintile of respondents across the four outcome metrics, while bottom performers were defined as the bottom quintile, with the three quintiles in between defined as middle performers. Respondents represented multiple sectors (including technology, financial services, healthcare, energy, and retail) and geographies (Americas, Asia, and Europe), providing a comprehensive view of how organizations are integrating AI into product development.

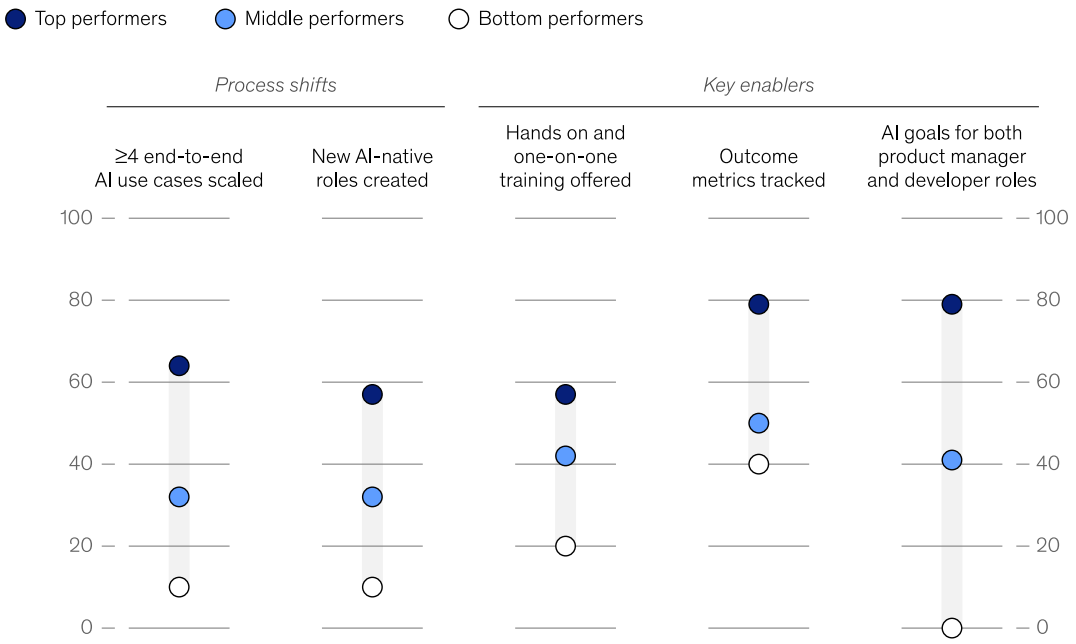
What sets AI software leaders apart

Leaders who actively shape mature AI adoption as part of software development are seeing material results, with a performance gap of 15 percentage points between top and bottom performers. High performance is characterized by higher artifact consistency and quality, shorter sprint cycles, smaller team sizes, and customer satisfaction scores. To understand what drives this outperformance, we examined this group's operating models more closely. We identified two key shifts, supported by three critical enablers, that consistently set them apart. Nearly two-thirds of the top performers used at least three of these five factors, compared with only 10 percent for their lower-performing peers (exhibit).

Exhibit 1

Higher-performing software teams use a combination of process shifts and key enablers to generate more impact from AI.

Share of respondents reporting practice in place at their organization, by performance,[1] %



[1]Top performers were defined as the top quintile of respondents across the 4 software development metrics (team productivity, customer experience, software quality, and time to market). Bottom performers were defined as the bottom quintile.
Source: McKinsey AI benchmark survey Aug 2025 (n = 100)

McKinsey & Company

Taken together, these five elements show how simply adopting AI tools is not enough. Companies must rethink how they structure teams and build software in an AI-forward world.

Two key shifts to unlock AI's full potential in software development

Top performers in AI-driven software development bridge the gap between experimentation and impact by embedding two mutually reinforcing practices across the product development life cycle (PDLC).

1. Prioritize end-to-end implementations of use cases across the PDLC

Top performers take a holistic approach, embedding AI across the entire development life cycle rather than limiting themselves to isolated use cases. They are six to seven times more likely than peers to scale to four or more use cases—from design and coding to testing, deployment, and adoption tracking. Nearly two-thirds of leaders report four or more use cases at scale, compared with just 10 percent of bottom performers.

Cursor, the fast-growing AI-native start-up, takes such a comprehensive approach. The Cursor team effectively operates as an internal lab for AI-driven engineering workflows, where teams trial solutions to their own pain points and "productize" those that gain traction. Developers combine AI agents, Bugbot support, and human review to expand coverage of software tasks with minimal disruption, while sprints balance the delivery of new features with process improvements. Engineers at Cursor set team commands, prompts, and rules at different levels of scope, whether for an individual file or the entire codebase, based on their preferences and the organization's design considerations, moving from sparse documentation to a "paved-path PDLC." The team uses the Plan Mode feature of their flagship coding tool to plot their changes before implementation, starting on customer feature requests in the editor or directly from their team communication tools.

During development, engineers collaborate with agents in real time, including through voice, to refactor code, ask questions about of their codebase, and build features. At the same time, they trigger background agents to handle other tasks—sometimes running multiple agents locally in parallel on the same task and later evaluating the best output directly within the editor. This setup allows the team to seamlessly switch between their current work and the tasks they have delegated to the background agent, often picking up where the agent left off. For iterative design and testing, engineers control and display a web browser within their editor to visualize changes during development, allowing them to audit accessibility, convert designs into code, and provide visual feedback during debugging. Bugbot, Cursor's AI code review tool, examines the resulting code before it is passed to other developers on the team for a final check, adding an extra layer of validation to any pre-production software. Overall, this new development workflow enables Cursor to increase their feature throughput with a lean team.[1]

[1] Based on interviews with Cursor team members.

2. Create AI-native roles within the PDLC

AI is increasingly taking on core engineering tasks such as refactoring, modernization, and testing—marking an evolution from assistive coding aids to fully integrated collaborators expanding across the PDLC. More than 90 percent of all software teams we surveyed use AI for these activities, saving an average of six hours per week. However, as engineering head count growth slows, developers are expected to pair technical fluency with product, design, and business understanding. This is enabled by more powerful tools with better orchestration, such as moving from basic autocomplete to reasoning-driven agents that plan tasks, supported by greater contextual information. For example, tools such as GitHub Copilot, Claude Code, and Google's Jules agent among others have evolved from simple inline completions to autonomously executing long-running multi-file refactoring and modernization tasks.

As part of this adoption, key roles are taking on new, AI-native responsibilities. Product managers, for instance, spend less time on feature delivery and more on design, prototyping, quality assurance (QA), and responsible AI implementation practices. For their part, software engineers focus more on full-stack fluency, structured communication of specs, and understanding of architectural and systems trade-offs. Both roles are building new AI-specific skills that augment their traditional strengths—skills in areas such as scale, security, and testing for developers, and strategy, customer insight, and governance for product managers. Looking ahead, many teams may operate as orchestrators of parallel and asynchronous AI agents, assigning workflows and shaping end-to-end logic together while continuously verifying outputs. Enterprises will increasingly create custom software on demand (for example, user-generated internal tools), making problem framing and intent specification critical skills and shifting organizations toward more product-oriented models.

The structure of the Cursor team reflects how AI-native roles require skill set shifts within the entire PDLC. Traditional boundaries between front end, back end, and QA have merged into broader full-stack responsibilities. Each release has a dedicated responsible individual (DRI) who coordinates development, testing, and bug resolution. Product managers test features internally, designers prototype directly in code, and business and data teams use the platform to query product data. Within machine learning (ML), infrastructure, and product, pods are small, and projects often span across teams. "Over the next decade, AI-assisted programming will let developers specify intent through a mix of formal programming languages and natural language, freeing them to focus on designing the logic of their software," says Michael Truell, CEO and cofounder of Cursor. "Some individual contributors may spend part of their time as engineering managers directing a 'junior' team of asynchronous agents—a new type of work that could demand an entirely new skill set. AI will likely play a role in code review and testing, and validation will accelerate by orders of magnitude."

Three critical enablers of success

As essential as they are, shifts in these two practices alone are not enough to capture AI's full value in software product development. Top performers reinforce these shifts with three critical

enablers—upskilling, impact measurement, and change management—that ensure adoption translates into sustained performance gains.

1. Upskilling: Invest in personalized, intensive training
While most companies are currently offering on-demand courses, those that invest in hands-on workshops and one-on-one coaching are much more likely to see measurable gains—57 percent of top performers versus only 20 percent of bottom performers. Upskilling engineers and product managers to use AI is not as simple as providing them with a tool. Breaking down problems to clearly communicate with a large language model (LLM)—prompt engineering—is just one example of the complexity that requires intensive training to raise the bar.

High-performing organizations design training that mirrors real development work—integrating AI into code reviews, sprint planning, and testing cycles—so teams learn to apply AI in live contexts, not simulations. They also personalize learning paths by role, focusing developers on prompt design and model evaluation, while helping product managers build literacy in model behavior, data governance, and responsible use.

Because tools are advancing so rapidly, training cannot be a one-off exercise. Static documentation or annual sessions quickly lose currency. Continuous and contextual coaching built into rituals such as retrospectives has become a key differentiator. Some leading firms have even established internal "AI guilds" or "centers of enablement" that curate new use cases, share best practices, and serve as on-demand mentors for project teams.

Ultimately, productivity depends as much on mindset and collaboration as on the tools and technology themselves. Teams that make learning part of delivery—treating every sprint as a chance to experiment and refine—are the ones consistently translating AI adoption into measurable business impact.

2. Impact measurement: Track outcomes—not just adoption
High-performing organizations know it's not sufficient to focus only on adoption metrics such as tool usage frequency or code acceptance rates. These outperformers track outcomes—monitoring quality improvements (79 percent) and speed gains (57 percent). By holding teams accountable for impact, leading organizations sustain momentum and adjust quickly when needed, while bottom performers focus solely on adoption metrics that, on their own, show little correlation with performance. As tooling evolves and allows for more capabilities (such as generating code from a design document), the "gold star" impact metrics will also evolve, forcing organizations to be flexible and adaptable. "Too often, companies measure AI's impact by counting how much code it produces rather than what that code achieves," says Tariq Shaukat, CEO of Sonar, maker of code quality analysis tools and solutions. "Lines of code or AI contribution percentages don't reveal whether the output is secure, maintainable, or even useful. The real progress comes from tracking how these tools help teams ship higher-quality, more reliable software—not just more of it."

Measuring AI impact effectively. Three steps can help build a robust measurement system:

— *Select meaningful metrics.* Define the outcomes that matter most, such as faster cycle times, higher-quality releases, and improved customer satisfaction. Avoid weak proxies like the percentage of code generated by AI, which offer little insight into real productivity. Overlay outcome metrics (such as productivity, speed, and quality) with input metrics (such as AI feature adoption or defect detection) to normalize the impact of progress over time.

— *Build integrated tracking.* Connect data across planning tools, code repositories, and AI usage logs to create a consistent view of performance. Integrated tracking helps identify bottlenecks in the development life cycle and ensures teams are making progress toward business outcomes.

— *Report insights regularly.* Continuously share findings with product, engineering, and business leaders. Regular reporting highlights successes, flags challenges early, and enables coordinated course corrections.

3. Change management: Align incentives with AI-enabled behaviors that drive impact
Top performers embed AI adoption directly into performance evaluations. Nearly eight in ten link gen-AI-related goals to both product manager and developer reviews, compared with just 10 percent of bottom performers for developers and none for product managers. By aligning individual objectives with the organization's AI strategy, companies create accountability and encourage employees to integrate AI into their day-to-day workflows.

Leading organizations focus incentives on the behaviors that drive impact—not just usage. Goals are framed around contributions such as identifying automation opportunities, improving velocity through AI-enabled testing, or enhancing quality via model-assisted code review. These behavior-based metrics create the foundation for impact while avoiding the pitfall of judging individuals on outcomes they cannot fully control.

High performers also expect teams to connect their AI-enabled work to broader outcomes such as productivity, quality, or customer experience. This builds awareness of impact without tying compensation to metrics outside individual influence. Embedding these expectations into performance systems moves companies beyond one-off mandates, making AI adoption a sustained organizational capability that continuously drives innovation and measurable value.

Moving toward true AI-driven value

Over the past couple of years, many enterprises have learned firsthand that generating true financial impact from incorporating AI into software product development requires organizations to make wholesale changes to their operating model, embracing critical new practices and enablers. This flexibility and ability to adapt are essential, given how rapidly the AI coding ecosystem is evolving, with new tools emerging every few months and model intelligence advancing rapidly. In just the past year, the AI coding benchmark from Artificial Analysis[2] has nearly doubled—from 30 points to 55 points—though this still trails the overall intelligence index

[2] A holistic benchmark of the top large language model coding benchmarks—including LiveCodeBench, SciCode, and Terminal-Bench Hard—evaluating models from such providers as Anthropic, Google, OpenAI, and xAI.

of all general models by 20 points, demonstrating that the AI coding tools still have room to grow smarter.[3]

Yet raw intelligence is only part of the story. Tools are becoming more powerful as they expand across the PDLC. With stronger orchestration and tighter system integration, they are progressing from simple autocompletion tools to hybrid, reasoning-driven agents that can plan tasks, call external tools, and even automatically simulate user testing through in-browser computer use, all supported by deeper context.

Organizations working to keep pace with these advances must also begin rethinking the structures and practices required to maximize their impact; implementing sustainable organizational change doesn't happen overnight, even as AI tools are accelerating at a breakneck pace. Software organizations that are leading the way in leveraging AI for real impact are using bold, end-to-end implementations that treat AI as a transformative catalyst for the entire development process. In addition to the practices and enablers we have outlined in this article, these outperformers typically follow three overarching steps: They *set ambitious goals* that unite leadership and energize the organization; they *develop a holistic blueprint for the future operating model*, tested and refined to fit the organization's context; and they *create a detailed road map* that redefines team structures, workflows, metrics, and incentives to unlock productivity at scale. Only by approaching AI in such a strategic, comprehensive way can software teams hope to harness its full potential as a force for innovation, efficiency, and value creation in software development.

Charlotte Relyea is a senior partner in McKinsey's New York office; **Martin Harrysson** is a senior partner in the Bay Area office, where **Nandita Bothra** is an associate partner and **Natasha Maniar** is a consultant; **Matt Linderman** is a partner in the Connecticut office; and **Jose Mario Pena** is a consultant in the Seattle office.

The authors wish to thank Aamer Baig, Courtney Lasserre, Daniel Ng, Diarra White, Eichel Choi, Gayatri Nair, Gursharan Bains, Jeremy Schneider, Klemens Hjartar, Megh Dholakia, Naveen Sastry, Prakhar Dixit, and Tara Balakrishnan for their contributions to this article.

This article was edited by Daniel Eisenberg, an executive editor in the New York office.

Copyright © 2025 McKinsey & Company. All rights reserved.

[3] Data provided by Artificial Analysis.