McKinsey & Company

Tech & AI Practice

The AI revolution in software development

If gen AI has a killer application, it's software development—one of the most profound shifts in the history of programming.

by Charlotte Relyea and Martin Harrysson

April 2026

‘Any sufficiently advanced technology is indistinguishable from magic.’

–Arthur C. Clarke

It’s 8 a.m., and the third floor of a bank in London comes alive as the day crew—three engineers shaking off the rain—steps into their office. Screens glow with activity. Logs scroll. The soft hum of their computers lingers in the air.

The AI agent teams—nearly a hundred of them—have just finished their shift, having spent the night refining a new cross-border payment system, testing failure paths, and shipping updates at a pace no human team could match.

The humans drop their bags and begin the daily ritual: a sprint review that now happens every morning, not every two weeks. Waiting for them is a neatly organized stream of AI-generated pull requests, test evidence, and risk flags—more progress in 12 hours than a traditional team might make in a month.

The job of the engineers isn’t to code so much as to steer, apply judgment to, and adjust priorities for the AI agents working for them. The engineers’ focus is much more on structuring agent tasks into precisely defined workflows, ensuring their activities are predictable and of high quality (for example, predefining the sequence of agent activities), and structuring templates for agentic output.

Sound like sci-fi? It’s not.

An agent factory for a large G-SIB[1] bank has successfully done this, including the new daily sprint cadence with humans. The results are staggering: 10 times the speed at half the cost. That’s a revolution!

Business questions this chapter will help you answer

- Are you thinking through the strategic implications of 20 times software development productivity?
- Is it important that your company be at the forefront of this revolution, or is it OK to be a follower?
- How do you really know if your organization is going up the software development productivity curve?

[1] Global Systemically Important Bank.

If gen AI has a killer application, it's software development. And its capabilities have grown exponentially over the past three years (exhibit).

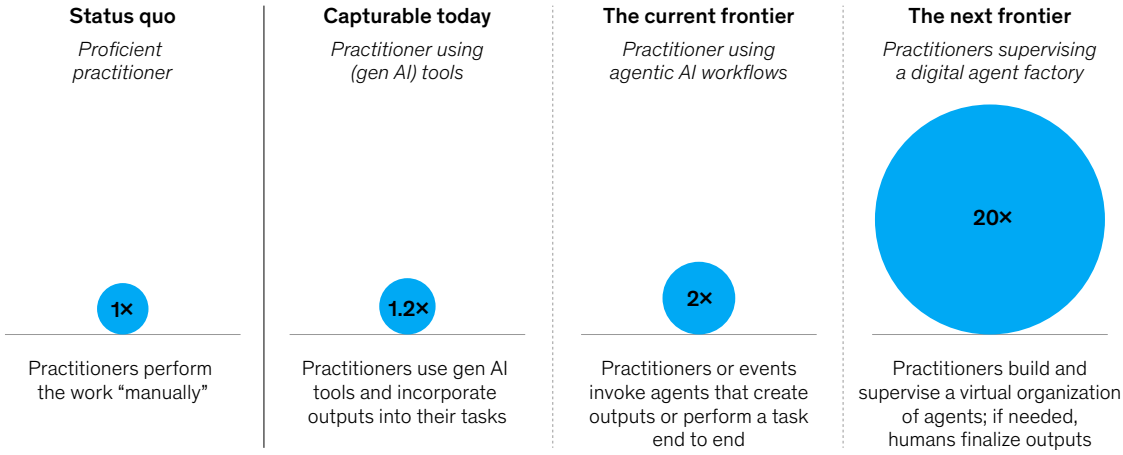
It's hard to overestimate the shift that's happening in software development. In essence, AI agents are running increasingly complex tasks and workflows (such as creating evidence provenance, running legal and cyber checks, testing counterfactuals, and both suggesting and making decisions). The role of humans is to declare high-level intent and boundaries, evaluate outputs, and react to agentic decisions and suggestions. This change is leading to smaller teams, much lower unit costs for software development, and much faster idea-to-impact cycle times.

To better appreciate the implications of this shift, it's helpful to understand the progression of gen AI's abilities in software development because that leap is starting to happen, albeit not as quickly, in other areas like law, consulting, marketing, HR, and finance.

Exhibit

A paradigm shift in software development is underway.

Raw productivity potential, by level of developer support, multiple



McKinsey & Company

The progress can be broken down into four levels of developer support:

— *Level 1: Developing without gen AI.* The software developer writes all the code alone. Quality is solid, but speed is limited by how fast one person can work.

— *Level 2: Speeding up individual tasks.* The developer writes a few lines, and the AI suggests the next ten, as though you had a super-fast pair programmer sitting beside you. AI provides a meaningful productivity boost.

— *Level 3: Automating entire steps in the workflow.* A developer describes a new feature to the AI agent in plain English. The AI generates the first version of the code, the tests, and the documentation automatically. The productivity boost is very substantial.

— *Level 4: Delivering entire applications.* A small team guides a coordinated system of AI agents that can deliver an entire application end to end—from design to code to testing to integration—raising only the decisions that truly require human judgment. The result is 20 times leverage: a few practitioners delivering what once required a large department.

Most companies are at Level 2 of this progression. Level 3 is increasingly being adopted as large language models (LLMs) have evolved from simple inline completion tools to autonomously executing long-running multifile refactoring and modernization tasks. Level 4 is largely experimental as of the writing of this book, though with promising developments already emerging.

Best practices for adopting AI in software development

McKinsey analyzed nearly 300 publicly traded companies to understand how AI is reshaping software development.[2] We found that a small group of top performers—roughly the top quintile—are achieving 16–30 percent improvements in productivity, time to market, and customer experience, along with 31–45 percent gains in software quality.

The key insight here is that simply giving developers AI tools does not meaningfully move the needle. The companies that unlock real value are those that rearchitect how they build software and deeply embed AI across the entire development life cycle—not just for coding.

They deploy multiple AI development use cases spanning ideation, requirements, design, coding, testing, deployment, and operations, enabling continuous acceleration and compounding benefits.

These organizations also make their development model AI-native, evolving roles, practices, and workflows so that humans act as orchestrators of AI agents. Developers shift from writing every line of code to supervising generation, validating architecture, and managing quality; product managers and designers take on more system-level thinking and integration of AI into features and experiences. It's a fundamental change in how teams work.

[2] "Unlocking the value of AI in software development," McKinsey, November 3, 2025.

Approach AI in software the same way you would for any strategic transformation—set direction, invest in skills, measure outcomes, and align incentives.

Behind these shifts are three critical enablers that determine success. Top performers:

1. Invest in serious upskilling, using hands-on workshops, real sprint simulations, and coaching rather than passive training.

2. Institutionalize tracking outcomes—release frequency, defect rates, customer experience—not just simple adoption metrics.

3. Reinforce change through aligned incentives and performance management. In fact, about 80 percent of top performers link gen AI goals to the evaluations of product managers and developers.

These enablers create accountability, accelerate learning, and help teams internalize new ways of working. Without them, organizations fall back into old habits, and AI's potential dissipates.

The implications are clear: AI can transform software development, but only for companies willing to rethink their operating model. Those that redesign workflows, roles, and governance around AI have the chance to create a true performance advantage.

A factory of AI agents: How does that work?

AI agents make it possible to run software development like a two-shift digital factory. Humans take the day shift, setting direction and enforcing quality. AI agents take the night shift, doing the heavy execution work—coding, testing, reviewing, documenting—inside a controlled, well-designed workflow. But reaching this point requires careful factory setup.

To begin with, the organization must prepare the environment in which agents will operate. Agents need structured requirements, clear user stories, and unambiguous acceptance criteria—they cannot infer business intent. They also need rich context about the system: domain knowledge, architecture diagrams, API contracts, data models, service boundaries, and nonfunctional expectations (such as performance and reliability). All of this is fed into the agent environment so the AI understands what it is building and why.

IBM's end-to-end software development transformation

When IBM embarked on its mission to redesign its software development process, it wasn't simply about writing better code. Leaders recognized that true transformation required looking holistically at the full development life cycle.

The initial rollout of a new AI-powered software development model was challenging. About 200 people were onboarded every other week; however, adoption was patchy, and tooling use and new behaviors didn't stick. Many tried AI tools, but when they initially didn't behave as expected, people reverted to their previous methods. After just a few months, it was clear the company had to take a different approach.

IBM knew, however, that the tool did, in fact, work and that it had the potential to take on an expanding range of tasks relevant for developers. There was then a commitment to a more comprehensive enablement program to truly upskill and support developers as they started embedding AI into their workflows.

This full-court press included assigning coaches to every team over the course of at least two sprints, having "bring your code in" office hours to resolve specific issues, and setting up and nurturing a lively Slack community where champions (those who knew how to use the AI tool well) answered questions quickly.

Over the course of about six months, IBM took more than 8,000 developers through this program. During that period, individual productivity increased significantly.

Once the factory is set up, the human team works the day shift. Their role is to decide what matters and convert that intent into agent-ready tasks. They refine user stories, translate features into specifications, break the work into well-scoped tasks, and define what "good" looks like. They provide architectural direction—explaining which modules can be touched, which should not be altered, and why. They set priorities, tune guardrails, and update tests for areas where agents have made mistakes. In short, humans move from typing code to directing, decomposing, and quality-controlling the work.

As evening comes, the night shift of AI agents takes over. A coordinated fleet of them performs multistep workflows: coding agents implement changes or refactor modules; test agents generate and run new test suites; QA agents identify regressions; security agents scan for vulnerabilities or leaked secrets; performance agents benchmark critical paths; and documentation agents rewrite and update API references and "what changed" summaries.

An orchestrator agent manages handoffs: if tests fail, it routes work back to a fix agent; if performance declines, it invokes a performance-checking agent; if a policy is violated, it halts the workflow. By morning, the factory has produced a set of ready-for-review pull requests, each containing code, tests, logs, analysis results, and a natural-language rationale.

The next day, the human team resumes the day shift by reviewing the output of the night. They examine the summaries, approve or refine code update requests, assess architectural fit, and give the AI new direction. They adjust priorities based on what the agents achieved overnight, tighten guardrails where needed, and mark more parts of the codebase as "safe to automate" as confidence grows.

In this model, software development becomes a continuous, high-speed loop rather than a two-week sprint cycle. The humans guide the system; the agents do the work; the engineering platform ensures safety and quality. The result is a factory that produces more, at higher quality, with humans focusing on the parts of the work that genuinely require expertise and judgment.

The new rhythm of work in an agent factory: Daytime is for judgment, design, and direction; nighttime is for execution, iteration, and improvement.

If you ask us, this is absolutely incredible. Success cases are still few and far between as of this writing, but breakthroughs are emerging. One large financial services firm, for example, has stood up this exact AI agent factory to develop a greenfield payment system and is improving productivity by 40–70 percent. LATAM Airlines has also experimented with a version of this and is delivering 50 percent increases in productivity (with smaller teams).

What does it take to run an AI agent factory like the one described above?

— *Don't skimp on the foundations.* Every successful implementation of AI agents has relied on strong foundations. LATAM highlights two in particular: a robust engineering platform that gives agents the tools and environments they need, and a product-oriented operating model where cross-functional teams already understand modern software engineering.

— *Invest in knowledge graphs.* Knowledge graphs are essential because they unify all the information inputs—code repositories, documents, and more—into a single structured network that shows how concepts, facts, and assets are connected.

— *Learn to break work into agent-ready tasks.* Humans need to develop the skill of decomposing larger features into small, well-scoped tasks with clear inputs, outputs, and acceptance criteria. This is what allows multiagent workflows to run safely. Without discrete, agent-ready work items, agents either stall or drift.

— *Master spec-driven development and context engineering.* Teams need to get very good at defining clear specifications—what the system should do, how it should behave, and how it will be tested. AI can generate code, but only when its instructions are crisp, structured, and complete.

  Equally important is giving agents the right context—architecture diagrams, data models, APIs, constraints, and business rules—so the AI can make correct decisions. Good AI output comes from good context, not clever wording.

— *Strengthen human judgment and review skills.* Humans become the editors-in-chief of the factory. They must review proposed updates, catch architectural drift, assess whether the agent's work matches intent, and decide when to tighten guardrails or adjust tests. This combination of product judgment, architectural understanding, and quality review remains fully human.

You can't "chat your way" to production-grade software. You need to master how to provide good instructions to AI agents.

— *Revisit performance expectations.* Human–agent productivity changes how teams operate. LATAM found one of the biggest challenges to adopting agentic AI was redeploying people into additional tasks as agents freed up time. Some companies reduce team size; others raise the bar on what should be delivered in a quarter. Either way, operating models must shift.

— *Monitor token consumption closely.* In a world where teams can spin up agents, which then create additional prompts or spawn subagents, token consumption can grow exponentially and lead to significant cost overruns (tokens are essentially processing units for LLMs). To counter this issue, build up your financial operations (FinOps) management to track and direct agent activity.

Running an AI agent factory is not about swapping humans for automation—it is about creating the conditions where humans and AI agents can work together at high speed and with quality. The human capabilities that sit on top—decomposing work, exercising judgment, tuning the system, and managing cost—are what will turn an agent factory from an experiment into a durable advantage.

What would happen if progress in software development productivity moved from the current frontier of two-times improvement to the new frontier of 20-times improvement? How would this change the world of business?

We know the road there might be a bit bumpy—any one of us can point to issues companies have been having even making modest productivity improvements with AI agents. But this world is coming, and senior executives should be thinking through scenarios and implications.

Here are a few to consider:

— *A 20-times lift in software development productivity turns established companies into continuous, real-time business improvers.* Customer journeys evolve weekly, not yearly, eliminating the inertia that has defined large incumbents.

— *Innovation becomes limited by imagination, not capacity.* New products, digital services, pricing engines, algorithms, and operational tools can be prototyped and tested in days or weeks.

What could your company do with a 20-times improvement in software development productivity?

— *Modernization stops being a massive program and becomes business as usual.* Legacy landscapes can be reworked in flight, removing the single biggest constraint on tech and AI transformations.

— *Companies with this 20-times capability begin to accelerate away from slower peers.* Faster releases, lower cost, better experiences, and tighter controls create a structural competitive advantage that compounds over time.

— *Operating leverage rises sharply as AI-driven productivity lowers the marginal cost of change.* Companies can ship more features, modernize more systems, and automate more workflows without adding headcount, widening the gap between output and cost.

Questions that matter

AI has the ability to unlock massive software development productivity gains, but it requires more than technology—bold leadership that can rethink workflows and operating models is also needed. Ask these questions of your C-suite:

Q1: Do we need to lead this 20x revolution or can we just follow our peers?

Q2: How will we track how AI in software development is improving the productivity and quality of our products/platforms?

Q3: How would our strategy change if the cost of development approaches zero?

Charlotte Relyea is a senior partner in McKinsey's New York office, and **Martin Harrysson** is a senior partner in the Bay Area office.