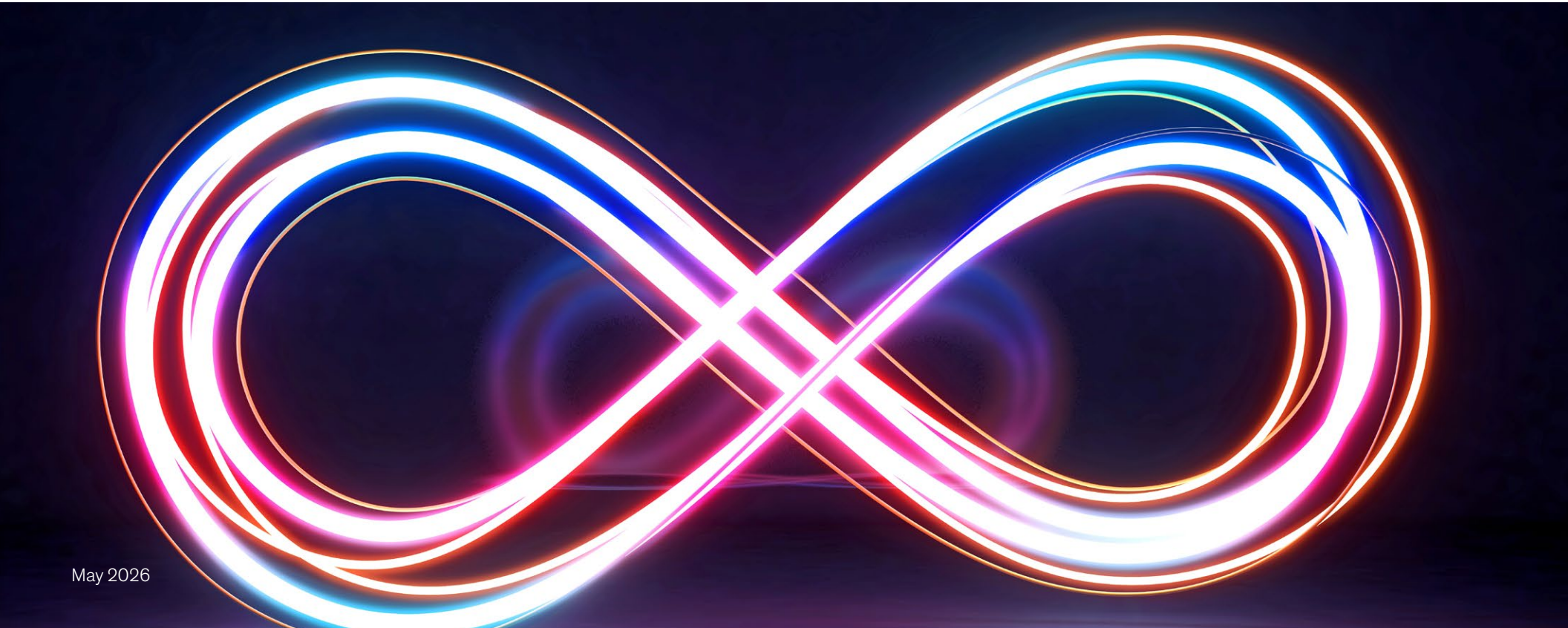
McKinsey & Company

McKinsey Technology

Rewiring software delivery for the agentic era

The way agentic AI is being used in software development is a harbinger for broader changes in the delivery model.

by Jared Moon, Rory Walsh, and Vito Di Leo
with Adam Thelwall



May 2026

At 9:00 a.m., a product owner logs in to review overnight progress on a solution her team is working on. She sees that a feature has moved from structured requirements to tested code. Edge cases are flagged. She notes that architecture dependencies have been validated. A concise summary outlines trade-offs and open decisions.

No one worked late. AI agents did.

By midmorning, the team is reviewing outputs, refining guardrails, and reprioritizing the backlog. By evening, the next structured inputs are queued up for the AI agents to work on over another overnight cycle.

This 24-hour work model is no longer theoretical. Leading organizations are already redesigning delivery around near-continuous execution. While the software delivery model is evolving quickly, multiple companies are already seeing it deliver threefold to fivefold improvements in productivity, with a 60 percent reduction in team size. Organizations are finding these gains not by just deploying AI agents but by rewiring the operating model so humans and agents can collaborate 24 hours a day.

While the software delivery model is evolving quickly, multiple companies are already seeing it deliver threefold to fivefold improvements in productivity.

The 24-hour sprint: Design for continuous throughput

Top firms are shifting toward a daily sprint model that blends human judgment with overnight agent execution—a significant reduction in the typical two-week-sprint cycle times. During the day, humans focus on reviewing outputs, resolving ambiguity, strengthening architectural guardrails, and aligning stakeholders. Increasingly, their role is less about producing artifacts and more about supervising and improving the system that produces them.

During the night, agents execute structured work at scale. Their tasks include enriching requirements, validating architecture, generating and testing code, and packaging outputs for review.

This model only works if a few practical foundations are in place. First, the business must have a clear vision of what needs to be built (for example, a product road map, or a standard to build from) so they can assess the agent-generated requirements for quality and alignment to that vision. The underlying technology environment then needs to be standard and consistent (for example, using common frameworks and modular architectures) so solutions can scale and components can be reused instead of reinvented each time.

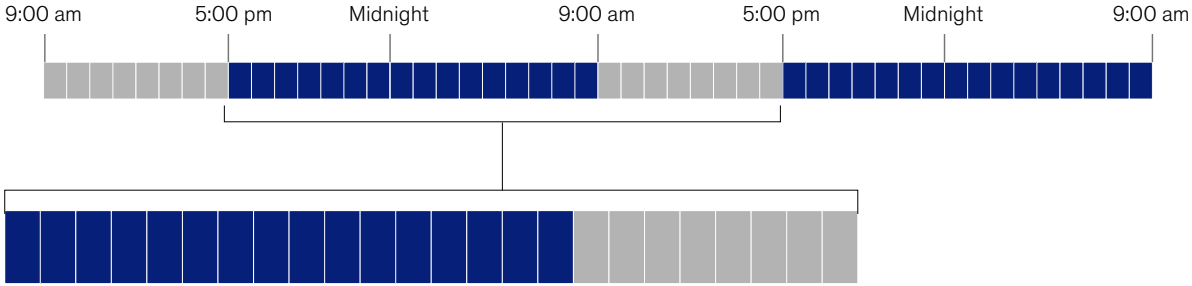
Third, the path from requirements to code must follow a standard structure so agents can reliably interpret inputs and produce predictable outputs across different projects.

And fourth, the same core stakeholders need to stay engaged across the value stream to avoid misalignment and constant rework. Without this level of consistency and clarity, agent output will be fragmented and difficult to trust.

Main takeaway: Continuous 24-hour delivery is achievable but only with architectural discipline and standardized workflows so agents can operate reliably at scale.

The AI-enabled operating model is based on daily sprints.

The way of working in the AI-enabled operating model (illustrative)



Night shift: 16 hours (led by a factory of agents)

Requirements	• Create the business requirements for the features requested by the humans
Architecture	• Check if the architecture is in place for the features • Set up the structure to build the features • Create the design for how to implement the features
Build and test	• Build and test a first version of the features • Write a report for the humans, with outcomes of the tests and recommendations

Day shift: 8 hours (humans supported by agents)

Sprint review/ demo	• Review agent output to identify gaps vs expectations and acceptance criteria
Spec-and-code working session	• Live pair review of critical code paths and AI traceability • Cross-functional sync (eg, legal/compliance inputs, design feedback)
Offline system optimization	• Refactor weak code flagged in morning session and fix outputs • Refine guardrails and quality standards for context, skills, prompts, and workflows; rerun the factory • Design improvements for next sprint
Sprint planning	• Refine inputs and instructions for the agentic factory (if needed) • Align with stakeholders on priorities for the upcoming night shift

McKinsey & Company

Extend automation to eliminate human handoffs

Traditional continuous integration and continuous delivery (CI/CD) automation focuses largely on testing and deployment. While those costs vary, our experience is that they can be as much as 30 percent of total technology spend. The majority of effort, concentrated in requirements through coding, remains manual and interpretation heavy. This is where friction accumulates and value plateaus.

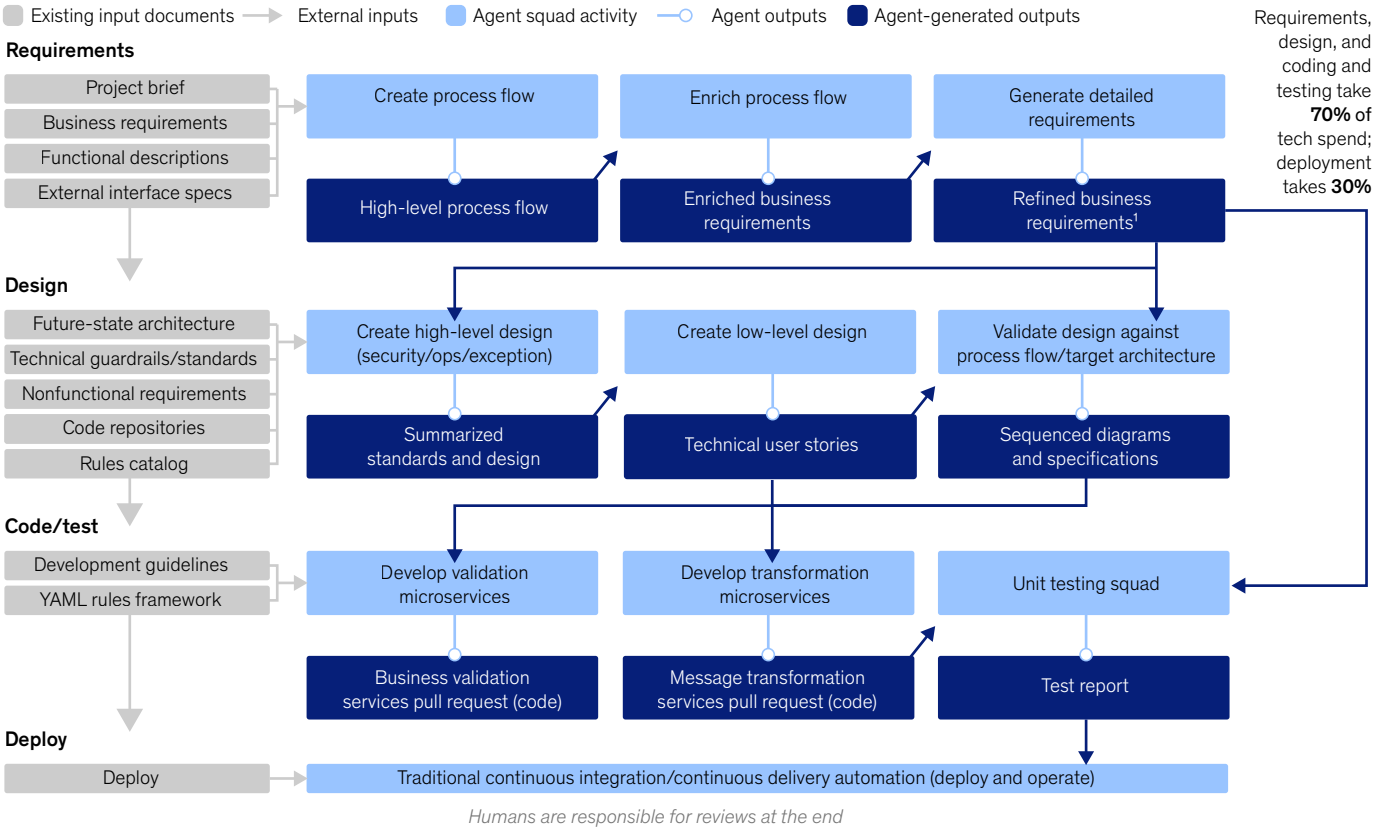
In most organizations, requirements, standards, architectural specifications, and user stories live across disconnected documents and tools. Each transition introduces ambiguity. Humans repeatedly translate intent from one artifact to another.

The agentic model removes this friction by structuring artifacts for machine-to-machine handoffs. Functional descriptions, nonfunctional requirements, guardrails, sequence diagrams, and repositories are codified in standardized, machine-readable formats. The pipeline can then run end to end in hours, with humans intervening only at defined review gates rather than acting as intermediaries.

Main takeaway: Scaling AI requires applying engineering practices to the development system itself, making the process repeatable and automating handoffs.

Rewire the product development life cycle to eliminate human handoffs.

Automated pipeline with agentic squads across requirements, design, build, and deployment



[1]Refined business requirements (1x doc per stage of the flow).
Source: "Software development cost: Complete 2026 budget guide," Boundev AI, Apr 2, 2026; McKinsey analysis

McKinsey & Company

Create a knowledge infrastructure to unlock agent autonomy

To produce accurate results, agent factories need organizational context and memory. Top businesses are building knowledge graphs that function as an AI memory layer across the software development life cycle (SDLC) for each domain. These graphs connect elements that agents need to make sense of, such as customer feedback, architecture decisions, design documents, tickets, GitHub activity, incident reports, and summarized compliance rules. The result is a semantically linked system (that is, a way for agents to understand what the data means so they can better perform their tasks).

The impact is transformative. Questions that once required weeks of interviews with multiple subject matter experts (SMEs) can be answered in minutes by a "librarian" agent drawing from structured institutional memory. Every decision becomes traceable. If a stakeholder asks why a feature was deprioritized, the answer can be linked directly to its source, such as customer survey data or usage analytics. Implicit tribal knowledge becomes explicit and explainable, reducing ramp-up time for new team members and strengthening governance.

Importantly, this should not begin with a grand, top-down ontology effort. The graph should evolve organically around priority domains and live programs, compounding value over time. As it scales, knowledge becomes production infrastructure, rather than static documentation, and a durable source of competitive advantage.

Main takeaway: Structured, connected knowledge is the foundation of agent autonomy. Treat your knowledge architecture as strategic infrastructure.

Knowledge graphs are the critical unlock to enable velocity and agent autonomy.

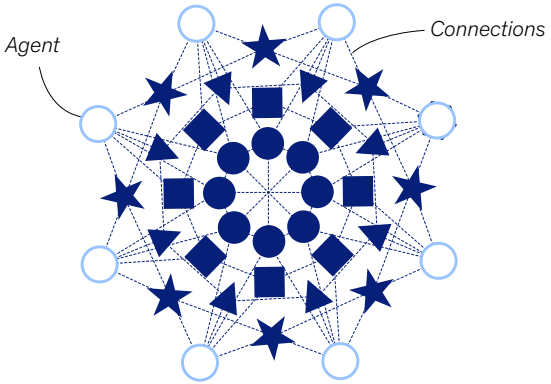
From manually connected context to an AI-enabled context layer (illustrative)

Humans manage context and draw connections manually	"Librarian" agent accesses knowledge on demand, handling complex multientity questions
Data is fragmented and disconnected. Making sense of it requires subject matter experts (SMEs) and lots of discussion	Data is organized with clear relationships between data entities within each domain that agents can understand

Example insight data by source

SharePoint	Results of previous surveys (eg, drop-off drivers)
Observability	Recent customer usage data and drop-off points
Jira	Features or user stories already planned for implementation
GitHub	Existing services that support onboarding or related capabilities
SMEs	Implicit SME knowledge on the journey (not formally documented)

Example knowledge graph



McKinsey & Company

Capture value: Resize teams and redesign the portfolio

The agentic SDLC can materially increase productivity because smaller teams can now do more work. Early implementations suggest larger teams of eight to 12 full-time equivalents (FTEs) could give way to smaller pods of highly skilled professionals supervising agent-driven execution. The result is compressed timelines and lower costs or increased capacity.

To capture the value, organizations should focus on three priorities. First is reskilling their people. While a central team needs the skills to develop and maintain "factories" of agents (ensuring standardization, compliance, best practice, et cetera), software engineers throughout the organization need to develop judgment, code review, and supervisory skills to manage the agents they work with. Roles shift away from manual coordination and testing toward architecture coherence, domain modeling, and AI supervision.

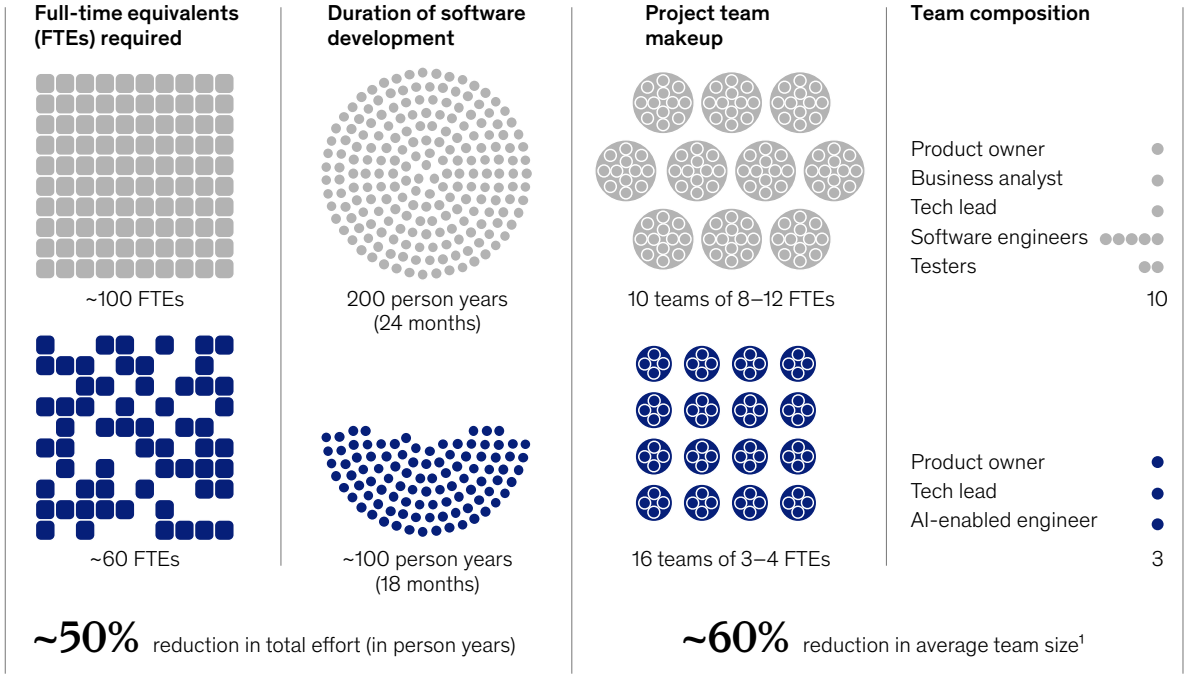
Second is ensuring the "outer loop" roles—support and compliance people in risk, legal, testing, and procurement—are part of the agentic development effort. A faster SDLC doesn't translate into faster progress if this doesn't happen. Agents and automation (for example, through policy as code) can help to ensure these controls don't become bottlenecks, while improving quality, consistency, completeness, and traceability. These controls should be baked in by design, rather than becoming a gatekeeper at the end of the process.

And third is redesigning how capacity is allocated so productivity improvements translate into new value. Freed capacity is often reinvested to accelerate road maps, modernize platforms, or launch new products.

Main takeaway: Productivity gains can be translated into structural portfolio changes. Resize teams and consciously redeploy capacity to capture full value.

Agentic software delivery requires smaller teams and less time.

Time and team size reductions (illustrative) Current delivery model Agentic software development life cycle



[1]Based on McKinsey experience and observation across multiple companies.

McKinsey & Company

Transformation should begin where impact is greatest. In most technology organizations, a small number of large programs account for the majority of total spend. Targeting these initiatives—whether legacy modernization efforts, brownfield rebuilds, or new product launches—maximizes visible impact and accelerates learning.

As agents take on execution at scale and produce code that is robust and consistently secure, human roles will concentrate in architecture, product judgment, and system design, making institutional knowledge and technical coherence decisive differentiators. Organizations that begin building these capabilities as part of a broader effort to rewire their operating model will not just move faster; they will redefine how software creates value.

Organizations that rewire their operating model will not just move faster; they will redefine how software creates value.

Jared Moon is a senior partner in McKinsey's London office, where **Adam Thelwall** is an associate partner; **Rory Walsh** is a partner in the Dublin office; and **Vito Di Leo** is a partner in the Zurich office.

The authors wish to thank Aishik Dhar, Amray Schwabe, Benjamin Schloesing, and Nikolaus Müller for their contributions to this article.

This article was edited by Barr Seitz, an editorial director in the New York office.

McKinsey & Company