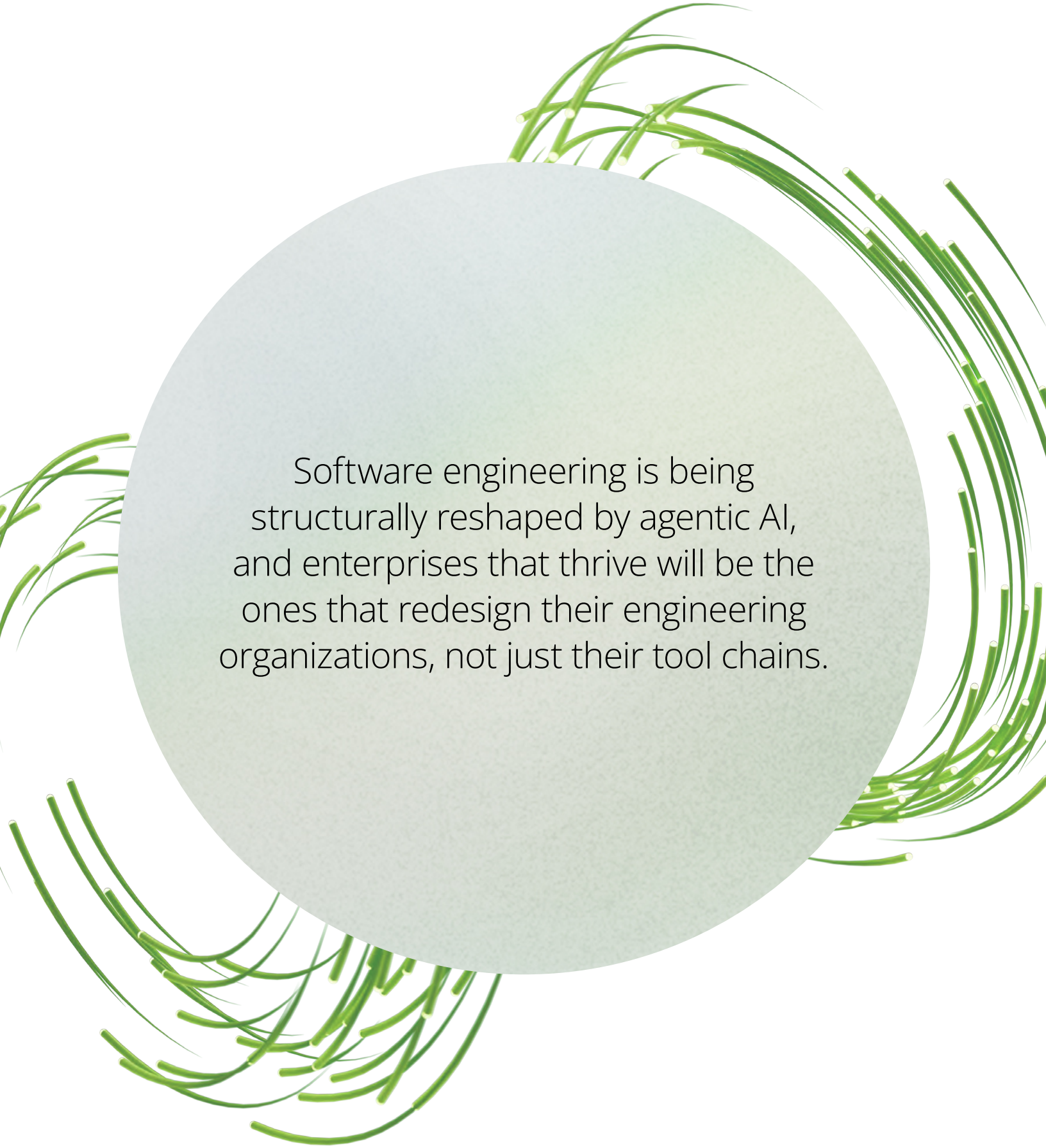
Deloitte.

Together makes progress

Future of Engineering: Agentic AI in software engineering — from SDLC to AO-DLC

May 2026



Software engineering is being structurally reshaped by agentic AI, and enterprises that thrive will be the ones that redesign their engineering organizations, not just their tool chains.

The inflection point:

Agentic AI changes everything

Software engineering reinvents itself regularly:

Agile replaced waterfall, microservices replaced monoliths, and cloud displaced on-premises infrastructure. Each transition took years to play out. Generative AI coupled with agentic AI represents the next such shift, except this one is extremely fast-paced.

The AI tools available between 2022–2025 offered one line or block of code generation capability at a time. Recent agentic AI systems can independently plan, make decisions, and execute multistep tasks across the software development life cycle. And the market is responding to this change with urgency. This urgency is a direct reaction to the staggering productivity gains expected through the use of these AI tools that can compress new product development (NPD) cycles by up to 50% and accelerate software development tasks by 30-40%[1]

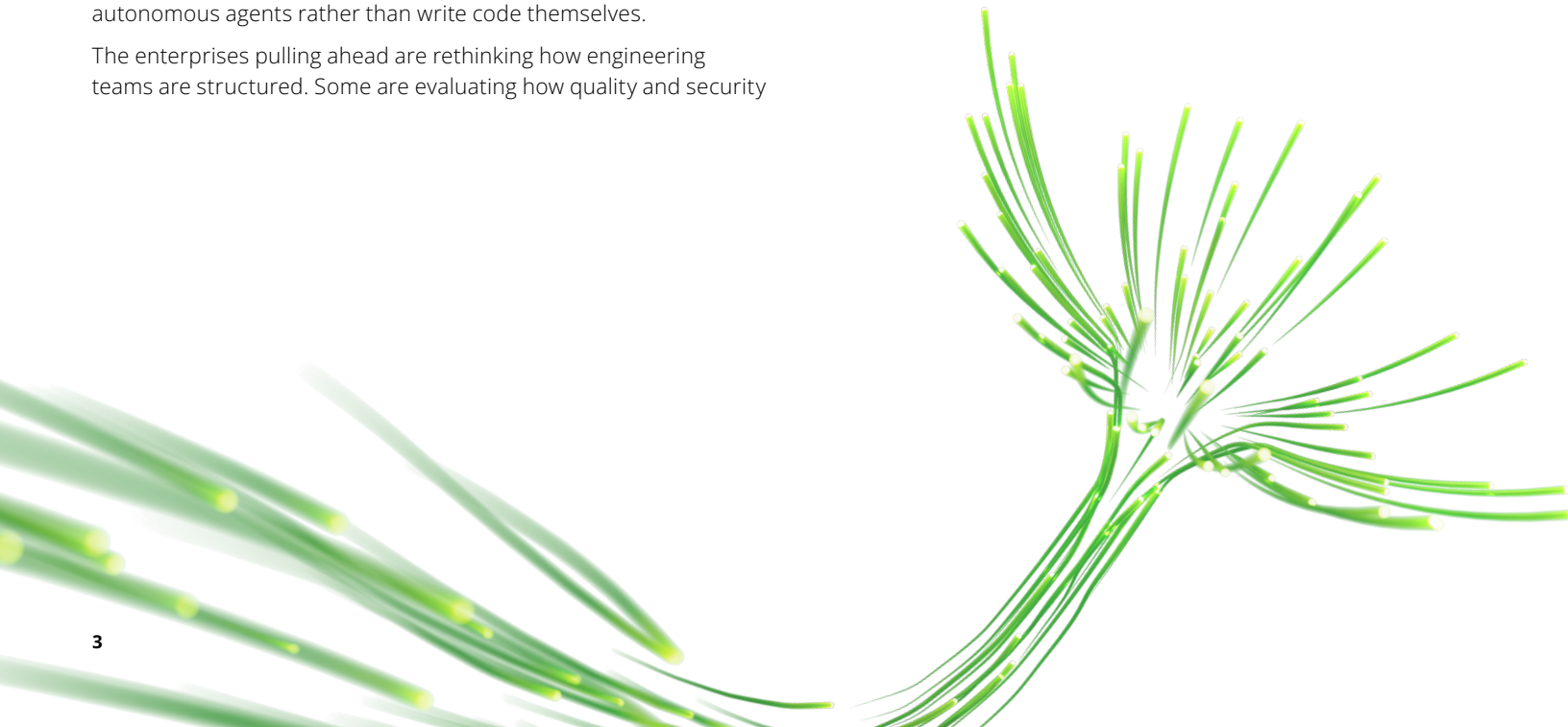
In just two years, the industry has moved from autocomplete-style code assistants to "vibe coding" (Collins Dictionary's Word of the Year for 2025),[2] where developers describe intent in natural language and AI generates functional code. By late 2025, the limitations of this approach had driven a shift toward "context engineering," the discipline of systematically managing how AI systems process context.[3] Now in 2026, the industry is entering the era of “agentic engineering,” where developers orchestrate autonomous agents rather than write code themselves.

The enterprises pulling ahead are rethinking how engineering teams are structured. Some are evaluating how quality and security are governed, while some are focused on how talent is developed. They are treating agentic AI not as an efficiency play but as an operating model transformation.

Every enterprise will be affected by this shift, and the ones that move deliberately in the next 12 months will likely set the terms while the rest may spend time adapting to standards.

This paper examines that transformation:

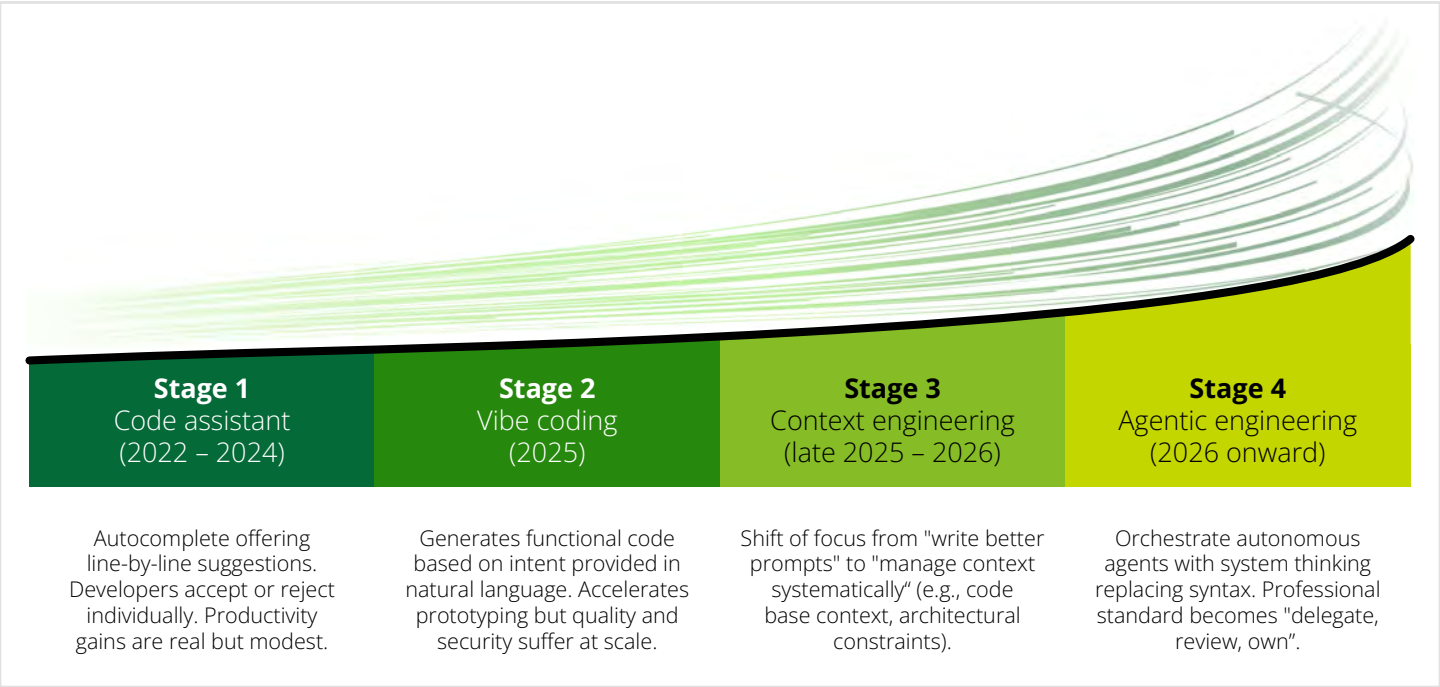
- How agentic AI is changing the SDLC
- Where the value lies
- How workforce roles are evolving
- What engineering-specific risks demand attention
- What leaders should do now



The maturation of AI in engineering

Three years ago AI in engineering meant a large language model (LLM) providing a code block for a particular problem or providing suggestions on how to resolve an issue. Today, entire development teams are being restructured around agent orchestration. The industry has moved through four distinct stages, each carrying different implications for how enterprises should structure their teams, tools and governance.

Figure 1. Stages of AI maturity in engineering



Source: Deloitte

In Deloitte's experience, most enterprises have deployed AI coding tools to developers but have only realized a fraction of potential value. We believe that most enterprises currently sit between stage 2 and stage 3, with a few in stage 1. Their developers have Generative AI coding tools and integrated development environments (IDE) but the surrounding infrastructure in terms of what context gets fed to agents, how AI-generated code gets reviewed, and who owns architectural consistency across agent-assisted commits is either absent or handled ad hoc by individual teams. The productivity gains evaporate due to the gap between tool deployment and organizational readiness. It is important to understand where your enterprise sits on this maturity curve as it will help you draw out a roadmap for transition to stage 4.

Transforming the SDLC:

From code writers to agent orchestrators and the beginning of AO-DLC (agent orchestrated development life cycle)

Through the past several decades, the software development life cycle has evolved to include clearly defined phases: requirements gathering, development, testing, deployment and maintenance. Agile methodologies made the process faster and more iterative. Agentic AI is now making it more intelligent, giving rise to what we call the "agent orchestrated development life cycle," or AO-DLC: an SDLC where autonomous agents handle execution across every phase while humans focus on architecture, validation and strategic oversight.

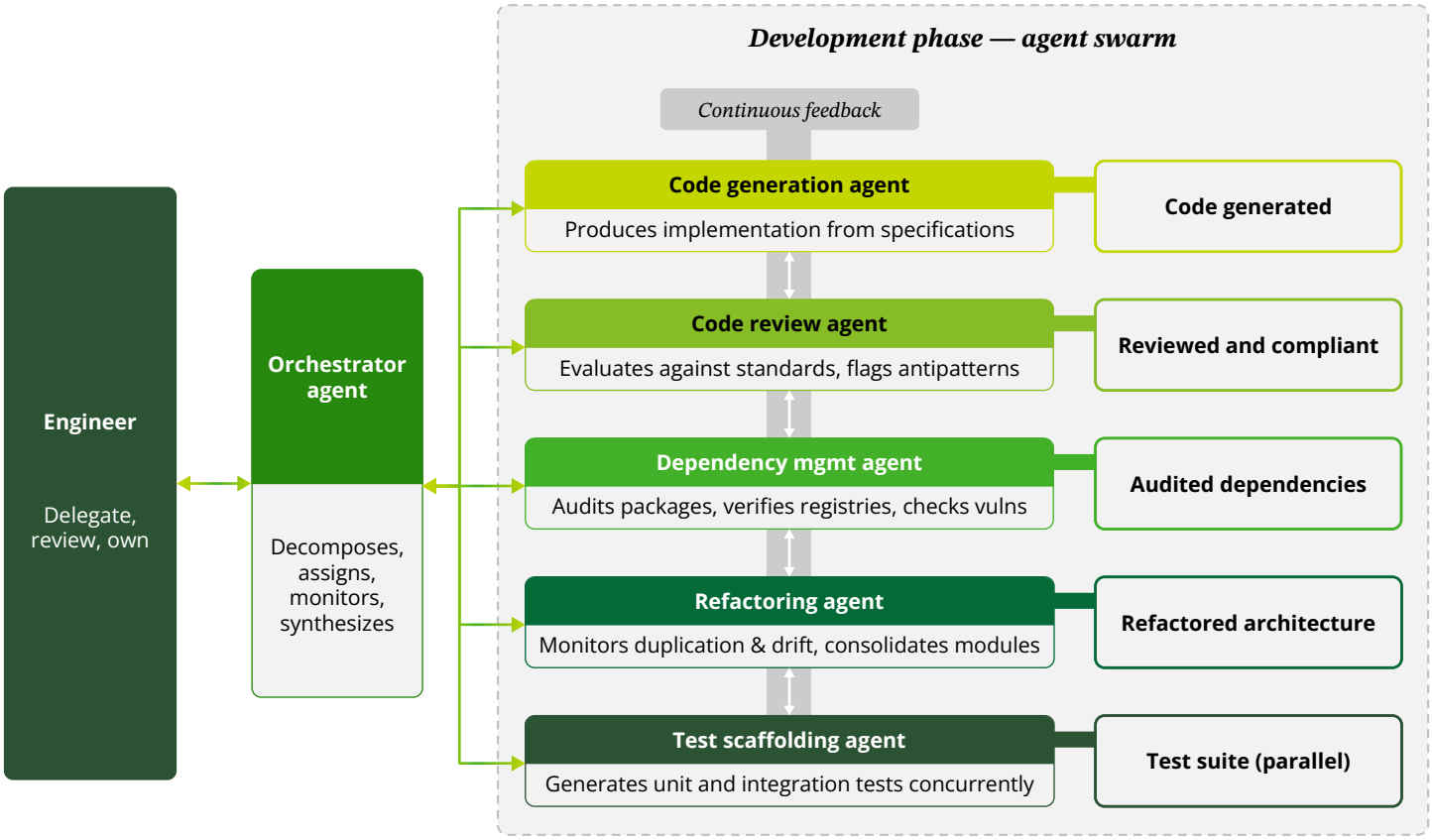
At the center of a mature AO-DLC sits a human engineer who directs an orchestrator agent. This orchestrator does not write code itself. Instead, it interprets high-level objectives, decomposes them into discrete tasks, assigns each task to the appropriate specialized agent, monitors progress, resolves conflicts between agents, and synthesizes results into an integrated output. The orchestrator is the conductor of what is increasingly being described as an "agent swarm": a coordinated fleet of purpose-built agents, each operating within a narrow scope but contributing to a shared goal.

Multiagent AO-DLC operations in practice

To understand the depth of this shift, consider what happens within a single phase of the SDLC, such as development. In a traditional setup, a developer writes code, runs it locally, debugs and submits a pull request. In an AO-DLC environment, the development phase alone may involve multiple agents working in parallel or sequence:

- A code generation agent produces implementation based on specifications handed down from the requirements agent.
- A code review agent evaluates the output against specific enterprise standards, flagging antipatterns of bad coding or design choices that likely will create later problems, and security concerns.
- A dependency management agent audits package selections, verifying that suggested libraries exist in official registries and are free of known vulnerabilities.
- A refactoring agent monitors for code duplication and structural drift, consolidating logic into reusable modules.
- A test scaffolding agent generates unit and integration tests in parallel with the code being written, so that validation happens concurrently rather than sequentially.

Figure 2. Illustrative representation of a development phase agent swarm



Source: Deloitte

This same pattern of decomposition applies across the AO-DLC life cycle. In requirements, separate agents may handle stakeholder input parsing, acceptance criteria generation, and specification conflict detection. In testing, agents may split across functional testing, security scanning, performance benchmarking and regression analysis. In deployment, agents may manage infrastructure provisioning, canary releases, rollback logic and production monitoring. The result is a system where no single agent is overloaded with context, and each operates within a scope where it can be highly effective. Furthermore, this structured, multiagent approach fosters predictability—with proper orchestration, the same prompt will not generate two vastly different, noncompliant code bases, but instead ensure outputs consistently adhere to organizational coding guidelines and architectural guardrails, thereby getting it right the first time.

Tooling to support this architecture

The tooling ecosystem is evolving to support this architecture. A new category of development environments, known as agentic IDEs, have emerged to provide the interface between human engineers and agent swarms. Unlike traditional editors that bolt AI onto existing workflows, agentic IDEs are built from the ground up around agent interaction.

These environments allow developers to describe tasks in natural language and delegate execution to agents that can plan, edit multiple files, run terminal commands, execute tests, and iterate until a goal is met. Key capabilities include multifile reasoning (understanding how changes in one module affect others across the repository), persistent memory (retaining context across sessions, including coding standards, past decisions and team conventions), parallel agent execution (running multiple agents simultaneously on different tasks using techniques such as Git work trees), and autonomous validation (agents testing their own output in sandbox environments before presenting results to the developer).

Realigned engineering focus

One of the most significant implications is that the AO-DLC is lowering the barrier between people who code to those who don't. Business analysts, product managers and domain experts are increasingly using agentic tools to build solutions that previously required professional engineering teams. This is accelerating the rise of what is called forward deployed engineers (FDEs): professionals who combine functional expertise with technical fluency to independently build, test and deploy agent-orchestrated solutions.

They are business-savvy professionals who understand the problem domain intimately and can now move at a pace that was previously reserved for dedicated engineering teams, using agents to bridge the gap between intent and implementation. These developers are shifting from writing code line by line to supervising multiple AI agents working concurrently, while FDEs handle an expanding share of solution delivery closer to the business. The skill that matters most in this environment is not syntax fluency but the ability to decompose problems, set architectural constraints, review agent output critically, and maintain system coherence across many simultaneous workstreams. Engineers need to increasingly focus on tying agent-built components into production-grade systems.

The business case:

Value creation

The business case for agentic AI in software engineering is promising and real.

Figure 3. There are several areas where value is generated:

Accelerated delivery:

By automating code generation, review, continuous integration and continuous delivery (CI/CD) orchestration, deployment and monitoring, agentic AI compresses end-to-end development timelines. Approximately 70% of developers using AI agents report reduced time spent on specific development tasks, and 69% agreeing to increased productivity.[4] Individual task speed improves, but the larger payoff is in coordination. When agents handle the handoffs between phases, bottlenecks that stall certain activities for days dissolve, shortening the overall timeline.

Rebalanced engineering economics:

When agents handle routine implementation, the economics of the engineering function shift. Engineers spend less time on boilerplate, scaffolding and repetitive tasks, and more time on architecture, validation and business logic. For enterprises facing talent constraints, this rebalancing can be the difference between delivering on a product roadmap and falling behind.

Quality as a governed outcome:

Agentic AI creates the infrastructure for continuous quality: agents that generate test cases, detect vulnerabilities, monitor application health and roll back releases based on real-time feedback. While quality improvement is not an automatic outcome of adoption, it is a governed outcome that requires deliberate investment in review standards, architectural oversight and human checkpoints.

None of this reduces the need for skilled human engineers. Rather, it raises the stakes with agents handling implementation—the quality of human judgment on architecture, business logic and system design becomes the decisive factor in whether the output is production-worthy or a liability. The ROI is achieved when developers focus their attention on what matters most: defining business logic, guiding architectural decisions and exercising thoughtful oversight.

To make the business case real, engineering leaders need to recognize that the magnitude of returns depends heavily on the maturity of governance and talent practices that surround adoption.

The workforce imperative:

Roles and skills

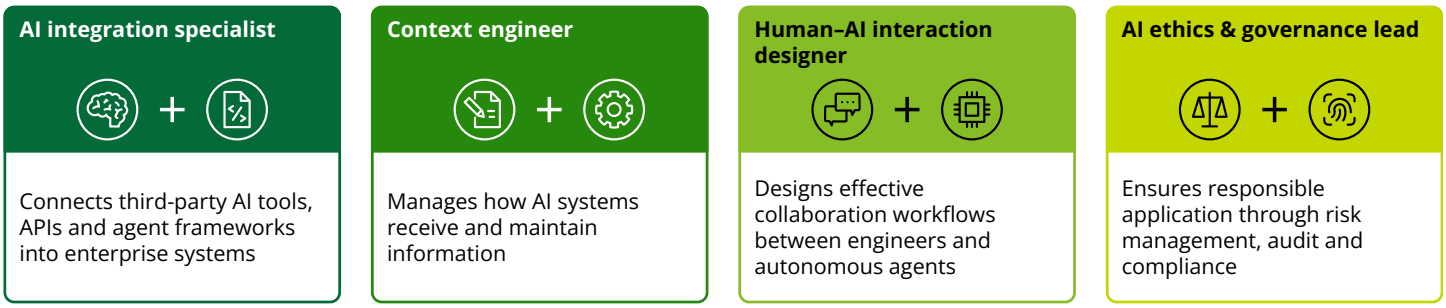
When engineers stop writing code and start orchestrating agents, everything downstream changes: what skills teams need, how roles are defined, and how performance is measured. Leaders who treat AO-DLC as a tooling change will find their teams unprepared while those who treat it as a workforce transformation will build the capability to sustain agentic AI at scale.

How roles are changing

Conventional SDLC roles are being redefined from separate, sequential activities into hybrid roles with concurrent AI collaboration. As examples,

- Software engineers increasingly inspect AI-generated code and solve AI limitations rather than writing from scratch.
- DevOps engineers incorporate AI assets into CI/CD pipelines.
- Test automation engineers review AI-generated test cases and manage AI-driven coverage.

Figure 4. This shift has also created entirely new specialized roles as seen in the image below.



Source: Deloitte

This means that engineering organizations are moving from a model of sequential, role-specific handoffs to one of concurrent, hybrid collaboration. Here, humans and agents work on the same problem simultaneously, each contributing what they do best—humans owning architecture, judgment calls and accountability, and agents owning speed, consistency and the ability to hold vast code bases in context.

New skills are in demand

The skills gap is real and widening. The capabilities that made engineers effective in a pre-agentic environment—e.g., syntax fluency, manual debugging and individual code authorship—are declining in relative importance. The capabilities that matter in an agentic environment are different:

- **Systems thinking:** Understanding how agent-built components interact across a complex architecture
- **Context engineering:** Providing agents with the right information, constraints, and conventions to produce quality output
- **Critical review at speed:** Evaluating large volumes of AI-generated code for correctness, security and architectural coherence
- **Agent orchestration:** Decomposing problems, assigning tasks to agents, and synthesizing results into production-ready systems

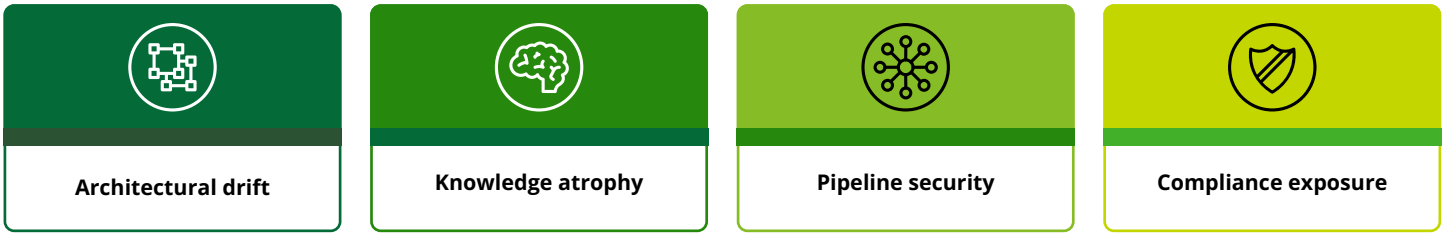
Enterprises that invest in upskilling now will build a workforce that can operate effectively at Stage 4 maturity. The risk of waiting is specific and measurable: engineers who are increasingly dependent on tools they do not fully understand, producing output they cannot fully validate.

The skills and role shift described above cannot be achieved through a single training program. It requires sustained investment in reskilling, mentorship structures that pair experienced engineers with AI-augmented workflows, and an organizational culture that treats learning as a continuous priority rather than a one-time event. Leadership should also attend to psychological safety: engineers whose roles are being fundamentally redefined need clarity about *their value* in the new model, not just instructions to adapt.

The risks of AO-DLC

While benefits described in the preceding sections are real, so are the risks. AO-DLC introduces risks specific to software engineering and distinct from the broader enterprise AI governance challenges most enterprises are already addressing. Proactively managing these engineering failure modes is what separates enterprises that scale successfully from those that accumulate unsustainable risk.

Figure 5. Risks induced by AO-DLC



Source: Deloitte

Risks induced by AO-DLC

Architectural drift: AI agents make design decisions implicitly, choosing patterns, adding dependencies, and structuring modules without complete awareness of the system's intended architecture. Across hundreds of commits, these microdecisions compound into what can be called "architectural drift." As an example, a GitClear analysis of over 211 million lines of code and commits showed 6.6% commits having duplicate code in 2024, up from 0.7% in 2020.[5] This outlines the criticality of enforcing architectural decision records and human-owned system design.

Knowledge atrophy: With developers now regularly using AI coding tools, the institutional knowledge required for debugging, incident response and architectural evolution is eroding over time. Over a long run, this could cause knowledge drain that AI cannot resolve through wider context. To combat that, we recommend enterprises require that engineers work without AI assistance on novel problems, and conduct regular solution walkthroughs and knowledge sharing to maintain core skills.

Security vulnerabilities in development pipelines: Agentic coding tools, especially those operating with elevated permissions across IDEs, CI/CD pipelines and code repositories, can result in critical vulnerabilities. It is important that enterprises enforce least-privilege agent permissions, create separate sandbox execution environments, and implement continuous authorization on every resource agents access.

Regulatory and compliance exposure: AI-generated code may lack audit trails, documented review processes and traceable decision logic, creating gaps in existing compliance frameworks. A SOC 2 audit will ask whether controls are in place over how code reaches production; if the answer is "an AI agent wrote it and no one reviewed it" that is a controls deficiency regardless of whether the regulation specifically mentions AI. Mandating documented review processes and AI-specific audit trails should be treated as foundational, not aspirational.

These are not theoretical governance concerns but measurable engineering failure modes that need to be addressed through a well-thought-through effort.

The path forward:

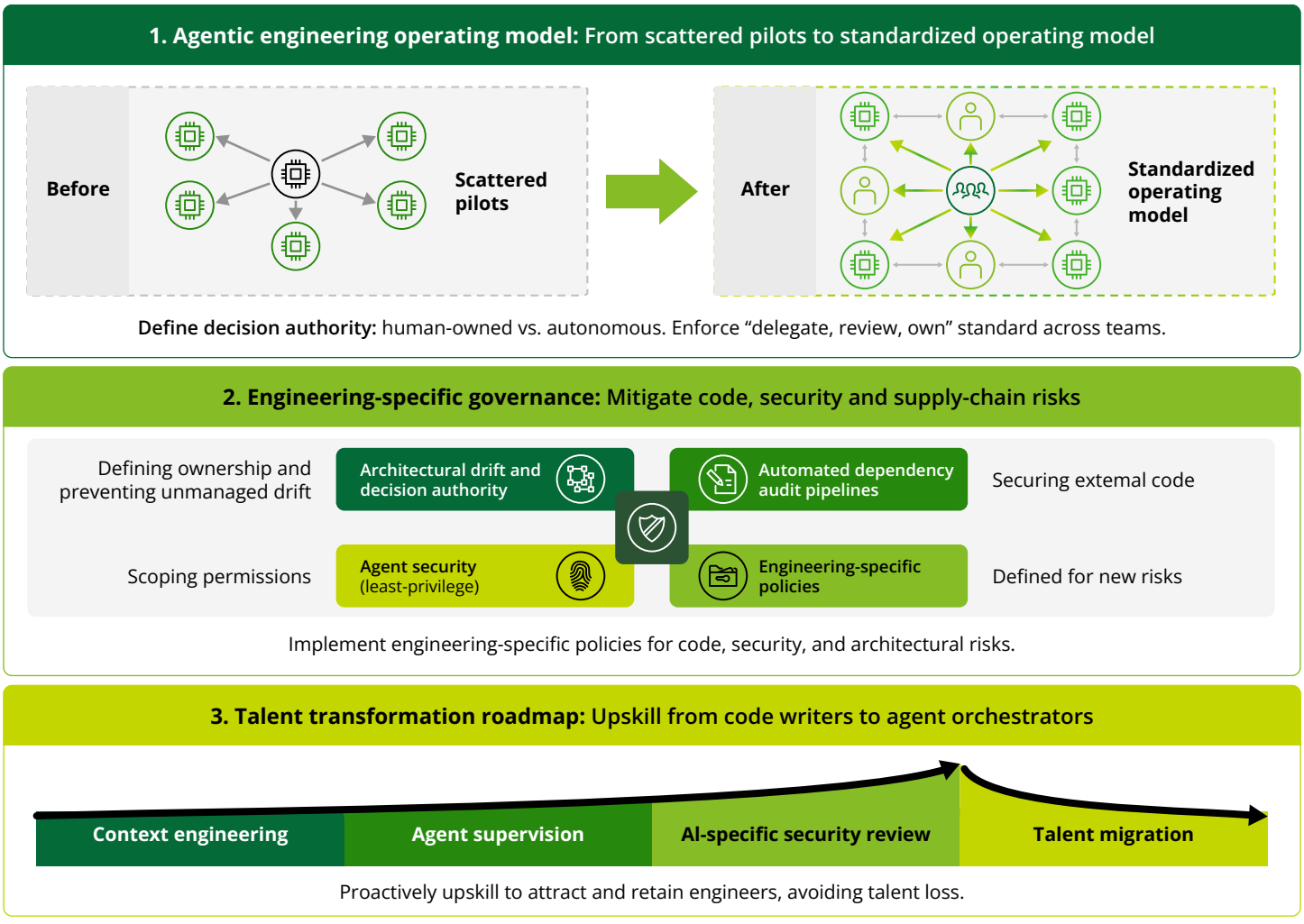
An executive action agenda

Agentic AI in software engineering is no longer a capability to monitor, but now is one to adopt. Yet the data suggests discipline matters more than speed.

Gartner® predicted that "over 40% of agentic AI projects will be canceled by end of 2027, due to escalating costs, unclear business value, or inadequate risk controls."[6] The failures will not come from moving too slowly but from adopting without the governance, talent strategy and architectural foundations to sustain it.

We believe that enterprises that treat AO-DLC as a new operating model will set the pace for their industries over the next decade, as their leaders focus on defining operating models, embedding governance and transforming talent.

Figure 6. Immediate priorities for agentic adoption in software engineering



Source: Deloitte

01. **Agentic engineering operating model:** Enterprises need to move beyond scattered pilots and define how agents will be orchestrated across the SDLC and who owns outcomes. As this paper has outlined, the engineering function is shifting from a model where individuals write code sequentially to one where teams orchestrate agent swarms across parallel workstreams. This requires clarity on decision authority: which architectural choices remain human-owned, where agents can operate autonomously, and how consistent accountability/review expectations are enforced across teams. Without a defined operating model, adoption will remain fragmented and ungovernable.
02. **Engineering-specific governance:** Generic AI governance frameworks are not sufficient for the risks outlined in this paper. Architectural drift and agent security in development pipelines require engineering-specific policies. Enterprises should define architectural decision authority, implement automated dependency audit pipelines, and scope agent permissions using least-privilege principles.
03. **Talent transformation roadmap:** The shift from code writers to agent orchestrators demands new skills that most engineering teams have not yet developed: context engineering (managing what AI systems know about the code base), agent supervision (reviewing and validating multiagent output at speed), and AI-specific security review (identifying the novel vulnerability classes introduced by agentic workflows). Enterprises that upskill proactively will attract and retain the engineers who matter while those who wait could find their talent migrating to organizations that have already made the shift.

The organizations that get AO-DLC right will be those that built the operating model, governance and talent foundations for successful scaling.

Navigating the shift, how Deloitte can help

Deloitte works with technology and business leaders to design and implement agentic AI strategies across the software development life cycle. Our teams bring deep experience in AI-native engineering, enterprise architecture, workforce transformation and AI governance to help enterprises move from experimentation to scaled deployment. Whether you are defining your agentic AI operating model, building governance frameworks for AI-generated code, transforming engineering talent and team structures, or embedding security and quality architecture into agent-orchestrated workflows, we can help you build the foundations that make this shift sustainable. To learn more, please contact the authors of this paper.

This article is part of Deloitte's Future of Engineering series, a collection of perspectives on how organizations are reimagining engineering to deliver impact at scale. Together, the series explores how leaders can combine AI and agentic ways of working with strong foundations across architecture, talent, quality and governance to drive lasting business outcomes.

Authors

Anantha Ramadas
Managing Director
Deloitte Consulting LLP
aramadas@deloitte.com

Jayanta Ghosh
Managing Director
Deloitte Consulting India Pvt. Ltd.
jaghosh@deloitte.com

Harshad Deshpande
Manager
Deloitte Consulting India Pvt. Ltd.
hadeshpande@deloitte.com

Acknowledgements

The authors would like to thank **Smriti Semwal** and **Rama Paruchuri** for their contributions to this article.

Endnotes

1. Deloitte, "AI-native workforce: Future of work and skills in engineering and product value chain," February 2026, accessed March 31, 2026,
2. Collins Dictionary, "Word of the Year 2025," November 2025, accessed March 27, 2026.
3. Ken Mugrage, MIT Technology Review, "From vibe coding to context engineering: 2025 in software development," MIT Technology Review, Thoughtworks' sponsored content, November 5, 2025, accessed March 27, 2026.
4. Stack Overflow, "2025 Developer survey: AI," accessed March 27, 2026.
5. GitClear, "AI Copilot Code Quality: 2025 Look Back at 12 Months of Data," 2024, accessed March 27, 2026, gated.
6. Gartner Press Release, "Gartner Predicts Over 40% of Agentic AI Projects Will Be Canceled by End of 2027," press release, June 25, 2025, accessed March 27, 2026.

Deloitte.