Capgemini

Modern Software Engineering

Contents

Building the next generation software organization

The discipline of software engineering is undergoing a foundational transformation, driven by the combined rise of AI agents and Platform Engineering. Modern software engineering redefines the way software is built, helping software teams to realize business value faster without compromising quality, security, or compliance.

This isn't just a story about tools or processes. It's as much about people and teams as it is about technology – requiring new roles, skills, and ways of working.

Software teams are increasingly delegating tasks to agentic software engineering platforms by planning, guiding, and controlling workflows that orchestrate AI agents and automated services. This enables teams to focus more on delivering business value and solving real problems, while improving compliance, quality, and security, and actively managing technical debt.

As a result, the workforce is evolving rapidly, with new skills, roles, team structures, and collaboration models emerging. Modern software engineering applies to multiple areas. It includes building custom software for new apps and business agents. Configuring and customizing packaged solutions. Derisking and accelerating legacy software modernizations. And securing and controlling the integration of new/modernized apps with core systems, data platforms and business agents within enterprise business processes.

Many organizations are now asking not just what the target model looks like, but how to bring their people along to operate in it. This paper explores how AI, software engineering platforms, and related workforce transformation are redefining what's possible for organizations. To move faster, smarter, and with greater impact, organizations must embrace these changes now.

Chapter 1

AI is reimagining the rules of software engineering

Software Engineering has been evolving for decades. Organizations have adopted and implemented multiple waves from agile, DevOps and product-centric ways of building software, through software lifecycle automation, to the recent rise of developer platform / Platform Engineering. And now the move to cloud native and composable software.

Since 2023, AI has been rewriting the rules of software engineering and redefining the software itself. This is no longer just about automating tasks. It's about establishing a genuine partnership between humans and AI. AI isn't just changing how software is built, delivered and operated. It's transforming the software itself.

Traditional and agentic software engineering approaches will coexist, but the transition to agentic software engineering is accelerating.

AI is no longer the future. It's now.

According to the 2025 DORA report, 90% of technology professionals now use AI in their daily work, with over 80% reporting increased productivity. This highlights the urgency for organizations to modernize their software engineering practices[1].

However, not all studies report such high levels of adoption or impact. For example, the 2025 Stack Overflow Developer Survey found that while 84% of developers are using or planning to use AI tools, only about half use them daily, and nearly half of respondents do not fully trust AI outputs[2]. Meanwhile, controlled studies have shown that experienced developers can sometimes be slower with AI due to the need for additional review and debugging[3].

The Capgemini Research Institute reinforces the overall trend, estimating that by 2026, more than four in five software professionals will leverage generative AI[4].

Organizations must act now to remain competitive and harness the full potential of AI-driven transformation.

A fast-maturing AI-powered tools market

In the last two years, the AI-powered tools market has matured at an unprecedented pace. It's evolved from specific generative AI-based assistants, such as code auto-completion or prompting user experience, to AI agents' orchestration engines supporting enterprise grade context engineering, human-in-the-loop and tools integration features, for all personas across the software lifecycle.

Today, agentic AI-powered tools have reached the maturity required for scaled usage and adoption in traditional and agentic software engineering contexts. They're now maturing towards uniformization, thanks to the rise and quick adoption of market standards in multiple agents' orchestration domains. These include agent skills, the approach to declarative context engineering, conversational human-in-the-loop, and dynamic tools / apps integration via the model context protocol.

Despite this momentum, only 27% of organizations have the necessary platforms and tools needed to fully harness generative AI, according to Capgemini Research Institute[5]. This gap poses significant risks, including shadow IT and security breaches, and highlights the importance of robust modern software engineering platforms and governance.

This readiness gap is not just technical – it is also organizational and human. It reflects missing skills, role clarity, and operating models, which slows adoption even where tooling exists.

[1] DORA | State of AI-assisted Software Development 2025
[2] www.allaboutai.com/resources/ai-statistics/ai-in-software-development/
[3] www.arxiv.org/abs/2507.09089
[4] Capgemini Research Institute: Turbocharging software with Gen AI
Gen AI in software - Capgemini
[5] Capgemini Research Institute found that only 27% of organizations have the necessary platforms and tools to harness generative AI's power, posing risks of shadow IT and security breaches.

AI agents are deeply impacting the software lifecycle

Generative AI has already established itself in software engineering, with code generation as its most common application. 71% of developers report using AI tools for this purpose[6]. These tools have delivered tangible results, driven by strong market pressure to adopt and experiment.

Agentic software engineering represents the next evolution - reshaping the software lifecycle through orchestrated collaboration between AI agents and software team roles, from business analysts to testers.

At Capgemini, we've identified nine core usage areas for agentic software engineering: knowledge and research, documentation, coding, reverse engineering, code transformation, DevOps automation, quality engineering and testing, application maintenance, and agile teams.

The effectiveness and impact of agentic software engineering is always shaped by the software organization context. Factors such as documentation and domain knowledge quality, codebase complexity and quality play a critical role. There is no silver bullet.

The challenge is no longer isolated productivity gains, but redesigning teams—defining what remains human-led, what is delegated to agents, and where accountability and validation sit.

Adoption patterns vary dramatically by company size, industry, and region

By company size:

Large enterprises lead adoption, with **41.2% of large EU enterprises** using AI versus only **11.2% of small firms.** Big Sur AI Among Fortune 100 companies, **90% have adopted GitHub Copilot.** Yahoo Finance Second Talent However, SMBs are closing the gap rapidly—U.S. small business AI usage doubled from **39% to 55%** between 2024 and 2025, and companies with 10-100 employees saw adoption jump from 47% to 68% in a single year.

By industry:

Technology companies lead with **85% AI adoption** and the highest acceptance rates for AI code suggestions (35%). Banking and financial services follow closely, representing **22% of the AI code tools market** with 80% of enterprises deploying Copilot. Second Talent Healthcare is the **fastest-growing segment** (CAGR 24.94%), driven by regulatory documentation and ambient scribing use cases. Grand View Research Professional services saw the biggest year-over-year adoption increase according to McKinsey.

By geography:

North America holds **36-39% market share,** Market.us anchored by $109.1 billion in private AI investment in the U.S. Asia Pacific is the fastest-growing region Grand View Research (19.8% CAGR), with **78% of APAC respondents** using AI weekly versus 72% globally. India leads with **57-92% adoption** depending on the survey, followed by China at **57-58%,** particularly strong in manufacturing. Europe shows more cautious adoption Grand View Research at **13.5% overall,** though **41% of large EU enterprises** now use AI - a gap attributed to GDPR compliance requirements.

[6] DORA | State of AI-assisted Software Development 2025

An alternative school of thought on productivity gains

While many reports tout impressive productivity boosts from AI, some analysts caution that adoption metrics can be misleading. Surveys often count any interaction with AI tools as 'adoption', even if usage is superficial, such as when junior developers rely on coding assistants for quick wins. This dynamic can inflate estimates, as juniors tend to report higher engagement without necessarily improving long-term maintainability or architectural quality.

In fact, studies[7] show that trust in AI outputs remains low among experienced developers, and many organizations are still in early maturity stages despite optimistic figures. The takeaway? Productivity claims should be viewed through a critical lens, considering skill-level bias and the gap between experimentation and true integration.

Changes in software engineering do not occur gradually, they happen in bursts. This is the theory of ***Punctuated Equilibrium*** in action. It says that when enabling conditions align, agentic software engineering capabilities leap forward, orchestrating better knowledge, clearer context and improved tooling.

Agentic software engineering spans custom software domains, from custom applications, package customization, and software data engineering to software product engineering. While its influence across the traditional lifecycle is undeniable, the journey has just begun. And the real impact will emerge as ways of working between humans and AI agents mature. Today, it's already safe to say that agentic software engineering is impacting the full software lifecycle, for all personas in a software team.

[7] Studies and source includes: The Fed - Measuring AI Uptake in the Workplace, How generative AI affects highly skilled workers | MIT Sloan, Seven Myths about AI and Productivity: What the Evidence Really Says | California Management Review

The new ways of doing software with AI agents

New practices and techniques, such as prompt engineering, context engineering, and agentic software engineering, are fundamentally changing how teams collaborate with AI agents to build and deliver software. These methods enable software teams to harness AI not just as a tool, but as an active collaborator in the development workflows. The result is a more dynamic, creative, and efficient way of building software where human expertise and AI capabilities are seamlessly integrated.

Developers now report a median of 16 months' experience with AI tools and methods and spend about a quarter of their workday interacting with AI[8]. Capgemini is also integrating AI into its own ways of working. Our framework is designed to embed AI across the software engineering lifecycle, from ideation to deployment. This transformation requires evolving culture and processes - not just adopting new tools.

At the forefront of this evolution is a concept that went viral in early 2025, when Andrej Karpathy described a shift in how he used AI to develop software in a social media post. He explained that he was increasingly accepting AI-generated code proposals without fully reviewing them, a practice he labelled "vibe coding." Since then, the term has been widely — and sometimes incorrectly — used to describe all forms of AI augmented software development. In reality, this covers a broad spectrum: from experimental, conversational approaches where humans and agents co create through intuition and improvisation, to more structured workflows involving formal design, planning, and specifications that AI systems use to generate code. Emerging ways of working in software engineering now rely even more on existing best practice but also encompass:

- **Prompt engineering:** initiates the process by crafting intentional inputs that guide the agent's behavior and set the direction of the task.
- **Context engineering:** provides the necessary scaffolding, surrounding the prompt with relevant information to enable continuity, precision, and deeper understanding.
- **Agentic software engineering:** delivers operational autonomy, allowing AI agents to execute tasks independently, interpret goals, and iterate with minimal human intervention.

- **Vibe coding:** mostly for expressive prototyping, where humans and agents co-create fluidly through intuition, improvisation, and shared rhythm.
- **Spec-driven design and development:** available through multiple market frameworks and kits.

The pace of innovation in this space is accelerating due to AI models and product developers reusing their own technology as they develop new versions and wholly new technologies.

We are starting to see the emergence of AI models that can be run locally on commodity developer laptops. These have capabilities and performance comparable with the models of just a couple of years ago, which at the time required the most powerful server-class GPUs. It's plausible that within a single hardware depreciation cycle (2026-2028), we'll see the release of open-source AI models that give developers device-local AI, with the equivalent of best-in-class 2025 AI-as-a-service performance.

Elsewhere, mainstream AI vendors will continue to develop very large models that push the limits of hardware. These will drive AI token consumption through the productization of semi and fully autonomous multi-agent systems, or swarms.

[8] DORA | State of AI-assisted Software Development 2025

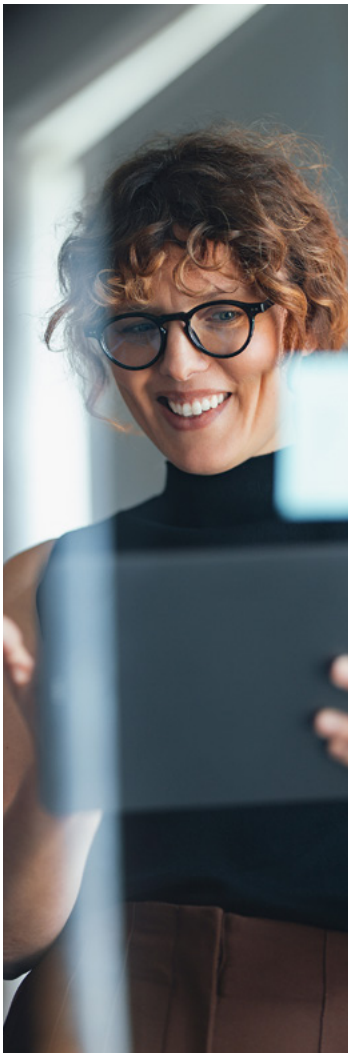
Harnessing these critical learnings now:

Modern software engineering demands a holistic shift across the lifecycle – something clearly reflected in the **graphic below** - from planning to maintenance, driven by AI integration and automation. Clear objectives, well-defined use cases, and accessible requirements are now critical. Not only for human teams but also for the AI agents that work within them.

Detailed architecture and domain-driven design have become essential to ensure adaptability and precision in AI-enabled environments. Agile practices are evolving as teams accelerate with AI tools. This requires new abstractions beyond traditional user stories to provide sufficient context for autonomous agents. Development workflows must embrace comprehensive build and test automation, as human-in-the-loop reviews increasingly create bottlenecks. Finally, maintenance is being reimagined with AI-driven automation for patching and observability, enabling near-zero technical debt and unlocking new value streams.

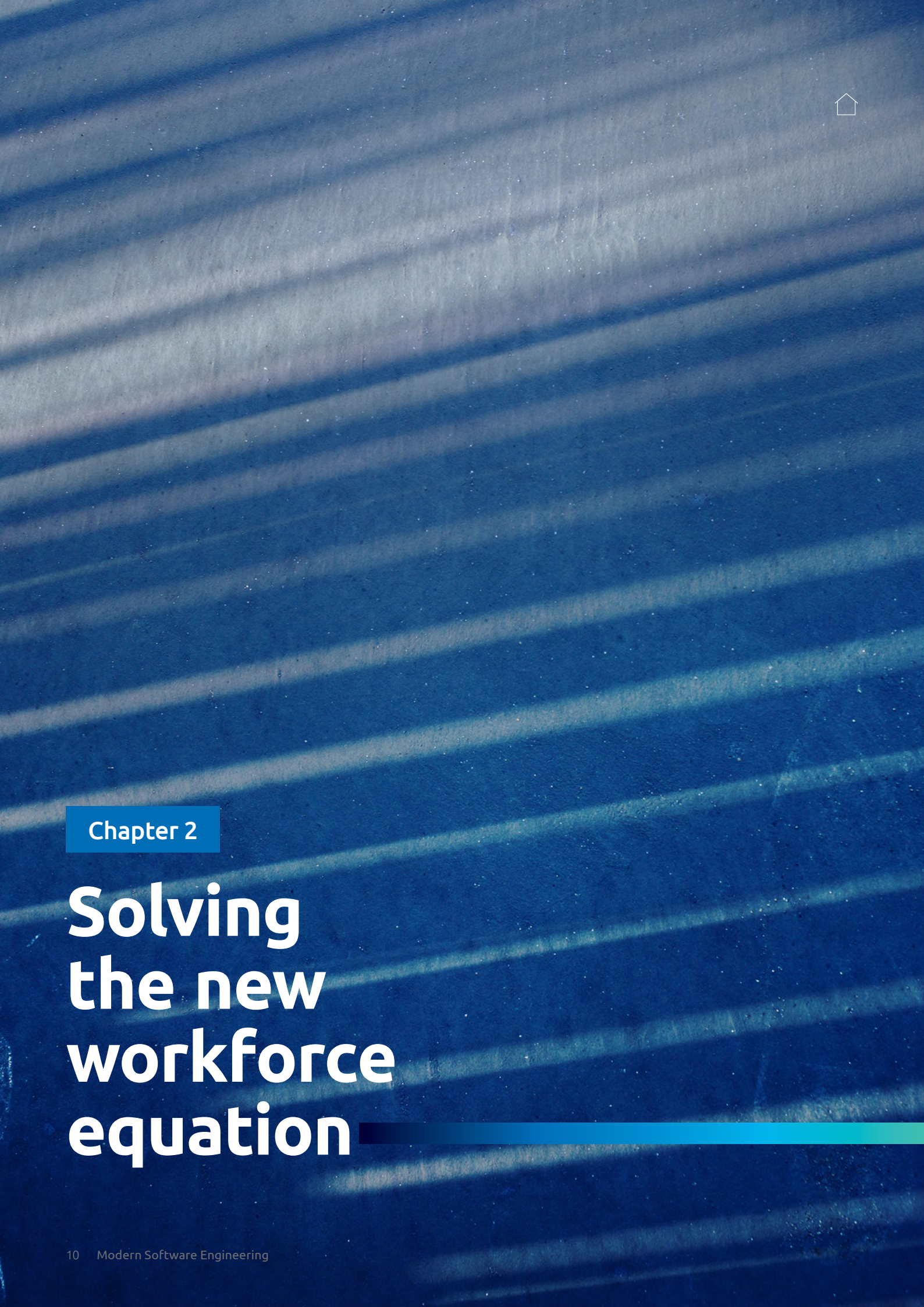
This transformation underscores that success in modern engineering hinges on context engineering, robust automation, and a redefined approach to collaboration between humans and intelligent systems.

Just as importantly, organizations must prepare their people to operate in this model. That means equipping teams to frame problems more explicitly, externalize tacit knowledge, work effectively with agents, and exercise stronger judgment over validation, escalation, and exception handling. Without this human capability shift, lifecycle automation alone will not translate into sustainable performance.



Planning	Design	Agile	Development	Maintenance
Clear objectives, use cases, requirements and success criteria are even more critical	Detailed software architecture, design and specification are more important than ever before	Teams change their ways of working as they accelerate when using AI tools	Vibe coding only works for experts that work alone, everything else needs process	AI agents can and should automate software patching and maintenance
Team composition matters: expertise in software engineering is essential	Design of the development process, prompts, AI agent & tool configuration is highly influential	As proficiency grows task size increases, reduced decomposition into granular tasks	Human-in-the-loop reviews of code are a bottleneck that impedes productivity	Near zero technical debt is a real prospect and will unlock new value streams
Discussions about requirements and the solution must be accessible to AI	Domain Driven Design is a practical & useful approach to designing with and for AI	User Stories are the wrong abstraction for coding with agents, (insufficient context)	Comprehensive build and test automation is essential	Comprehensive and robust observability is essential

Skills & Competency | Context Engineering | Methods | Automation



Chapter 2

Solving the new workforce equation

The emergence of agentic software engineering marks a true paradigm shift. Organizations are being pushed beyond simple deterministic automation pipelines to a new model where humans and AI agents work collaboratively in software teams. This transformation is not just about increasing speed. It's about fundamentally redefining how work is accomplished, how teams are structured, and how value is created. The true bottleneck in scaling agentic software engineering is not only technical maturity, but an organization's ability to redesign work, clarify accountability, and enable people to operate effectively alongside intelligent agents.

The human factor as the max factor

Peter Naur's 1985 paper[9] '*Programming as Theory Building*, reframed software development as the creation of a shared mental model - held not just in code, but in the minds of developers. This aligns with the concept of tacit knowledge, which research shows represents **70–80% of organizational knowledge, with 42% of job-critical knowledge existing only in individuals' heads.**

In many organizations, this is the real constraint. It's not the availability of AI, but the fact that critical knowledge remains implicit - embedded in experience, informal practices, and unspoken judgment.

This creates a challenge for human–AI collaboration. When tacit knowledge is not externalized, AI systems cannot fully understand design intent or context. Code and documentation capture outputs, but rarely the underlying rationale or trade-offs. As a result, AI operates on incomplete context—leading to weaker outputs, brittle automation, and misaligned decisions.

In short:

- Human insight remains fundamental to software development
- Most of that insight (70–80%) is tacit and not directly accessible to machines
- Externalizing that knowledge is essential for AI to reliably augment, rather than undermine, engineering outcomes

From Doing to Governing: the evolution of human roles & mindsets

Agentic software engineering is fundamentally changing what it means to be part of a software team. As AI agents take on a growing share of execution, human roles are shifting toward higher-value contribution - shaping intent, setting context, orchestrating workflows, validating outcomes, and governing performance. This is not simply a transfer of tasks from people to machines. It is a redefinition of how human expertise creates value across the software lifecycle.

This shift follows a clear progression in how roles evolve:

- **From doing to guiding:** Humans move beyond manual execution toward defining objectives, framing problems, and providing the context agents need to operate effectively.
- **From guiding to collaborating:** Humans and AI work together in iterative workflows with people shaping direction, interpreting outputs, and refining results as work progresses.
- **From collaborating to directing:** Human experts set objectives, constraints, and quality standards, ensuring that AI-powered execution remains aligned with architecture, business priorities, security, and compliance requirements..
- **From directing to governing:** As autonomy increases, human responsibility becomes more strategic – designing guardrails, manage exceptions, arbitrating trade-offs, and retaining accountability for the integrity and impact of outcomes.

[9] Naur, P. (1985). Programming as Theory Building. Microprocessing and Microprogramming, 15(5), 253-261.

This progression also reshapes teams. As execution becomes increasingly automated, competitive advantage shifts toward how organizations redesign teams, rebalance seniority, and define where human judgment, escalation, and governance remain essential. Roles become more fluid, placing a premium not only on technical capability but also on adaptability, strategic oversight, and the ability to collaborate effectively with autonomous agents.

This evolution is not only a shift in activity, but also in accountability and seniority mix. As AI agents take on more execution, junior roles evolve towards faster capability building through AI-assisted delivery, while senior engineers, architects and leads take on expanded responsibilities for design integrity, governance and validation. Clear human-in-the-loop accountability becomes essential, ensuring that ownership of quality, security and business outcomes remains explicitly defined, even in highly automated workflows.

This shift demands stronger judgment, adaptability, and the ability to guide multiple possible outcomes - not just execute predefined tasks.

The organizations that succeed will be those that treat this as a workforce design challenge as much as a technology one, building the roles, capabilities, and accountability models required for humans and intelligent agents to perform effectively together.

Coders become collaborative conductors

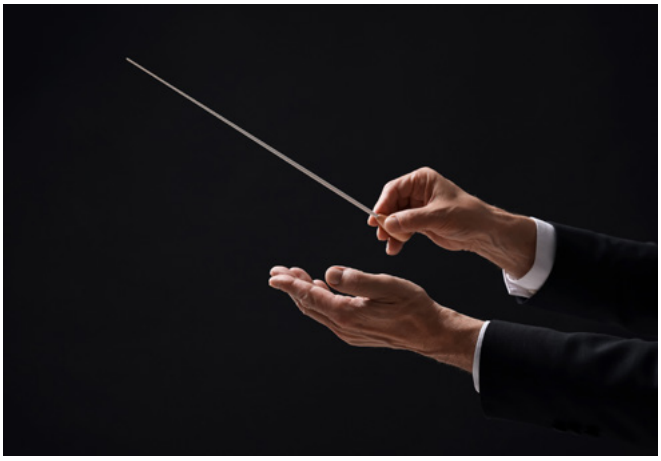
In leading organizations, software engineers are producing more software but, paradoxically, they're writing less and less code. Instead, they're orchestrating teams of AI agents to write code for them. For example, a modern engineering lead might design the overall system architecture, set the objectives and guardrails for AI agents, and oversee the quality and compliance of outputs, rather than manually implementing every feature.

This shift is visible in hiring trends. Demand for generative AI skills has surged, with U.S. job postings mentioning these skills increasing by 323% in just one year[10]. Companies investing in upskilling and reskilling their teams are seeing faster delivery, improved quality, and greater adaptability.

This is what agentic software engineering looks like in practice. Humans moving from direct execution to strategic oversight, guiding, validating, and governing AI-powered workflows to deliver business value at scale.

Key competencies for the agentic era:

- Strategic oversight and judgment
- Designing, orchestrating and validating agentic workflows
- Context engineering and problem framing
- Ethical alignment, risk management and regulatory compliance
- Continuous learning and adaptation
- Ability to operate across different levels of AI maturity, from assisted to semi-autonomous and fully agentic systems



[10] DORA | State of AI-assisted Software Development 2025

Delegating to AI is not abdication. It's a call for greater human responsibility. The more we empower AI, the more essential it becomes for humans to govern, orchestrate, and innovate. Organizations that invest in upskilling and reskilling their teams for these new roles will be best positioned to harness the full potential of agentic software engineering. This requires a structured approach to upskilling and reskilling at scale, ensuring teams can continuously evolve as roles, tools and levels of autonomy mature.

Skills progression and proficiency models: From AI-assisted to agentic delivery

Transforming a software organization into a modern, agentic software engineering model requires deliberate evolution across the three dimensions: **human / agent collaboration and teams, new methods and workflows, and the required critical foundations that enable scale.**

Most organizations typically started by infusing AI assistance to each software team role, with first outcomes in terms of individual productivity, at least in the code-centric workflows, but without any significant difference in software lifecycle workflows.

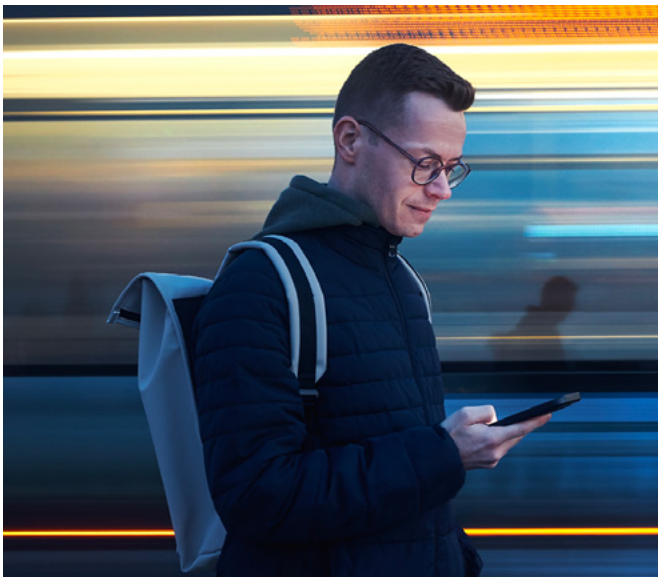
Organizations are now progressively infusing **human / agent collaboration** in software teams, where skilled software engineers are pairing with orchestrated agents across more software engineering workflows. Organizations understand and monitor the skills and effort needed for agents' control, for their output validation and rework. They identify, in their context, the optimal mix of skills, seniority and roles in the new software teams.

In parallel they're implementing an industrialized **agentic software engineering framework** combining end-to-end practices and **methods** applied in agentic software engineering **workflows**, across the software lifecycle.

Finally, they identify and experiment with the **critical foundations** required for successful deployment of agentic enterprise-grade software engineering at scale. These are cost control, security and guardrails, performance and adoption measurement, and the need for (agentic) Platform Engineering. Most importantly, they lay the foundations for agents' output control and validation, including quality, security, compliance, and business objectives, throughout the software engineering lifecycle.

Trust is nuanced...

Despite widespread adoption, trust in AI-generated code remains nuanced. According to the DORA 2025 report, 70% of developers express confidence in AI-generated code, but 30% report little to no trust, emphasizing the ongoing need for human validation and oversight[11].



[11] DORA | State of AI-assisted Software Development 2025

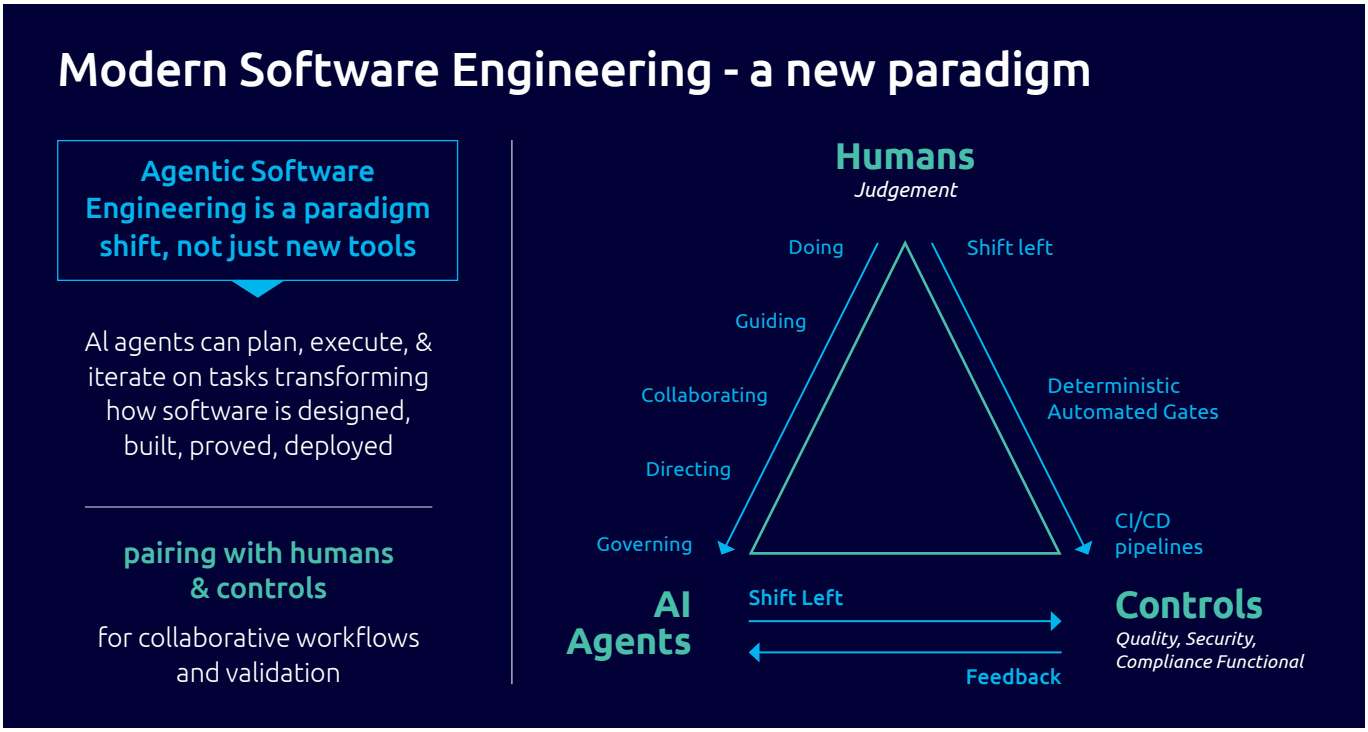
Control and validation: the foundation of human–agent collaboration

Agentic software engineering introduces a bold shift in the power dynamic. AI agents are no longer just assistants; they're fast-becoming active participants in software delivery. But with greater autonomy comes greater responsibility. Delegating tasks to AI doesn't eliminate the need for control and validation, it amplifies it.

As AI agents plan, execute and iterate on tasks, software teams must ensure outputs meet quality, security, compliance, and business objectives. This requires a three-pronged approach:

- **Human oversight:** Lead engineers and domain experts guide agents, validate outputs, and rework where necessary. They define objectives, set guardrails and ensure workflows align with client realities.
- **Automated controls:** Embedded into delivery pipelines, these gates enforce standards at scale, covering code quality, security checks, regulatory compliance, and functional requirements. They act as non-negotiable checkpoints that keep speed from compromising stability.
- **Autonomous agents:** As agents gain more independence, they must operate within well-defined boundaries, leveraging embedded governance rules, self-check mechanisms, and escalation paths back to humans when exceptions occur. Autonomy without control is risk. Autonomy with structured validation is innovation at scale.

The Bottom Line: Control and validation are essential to scale AI safely.

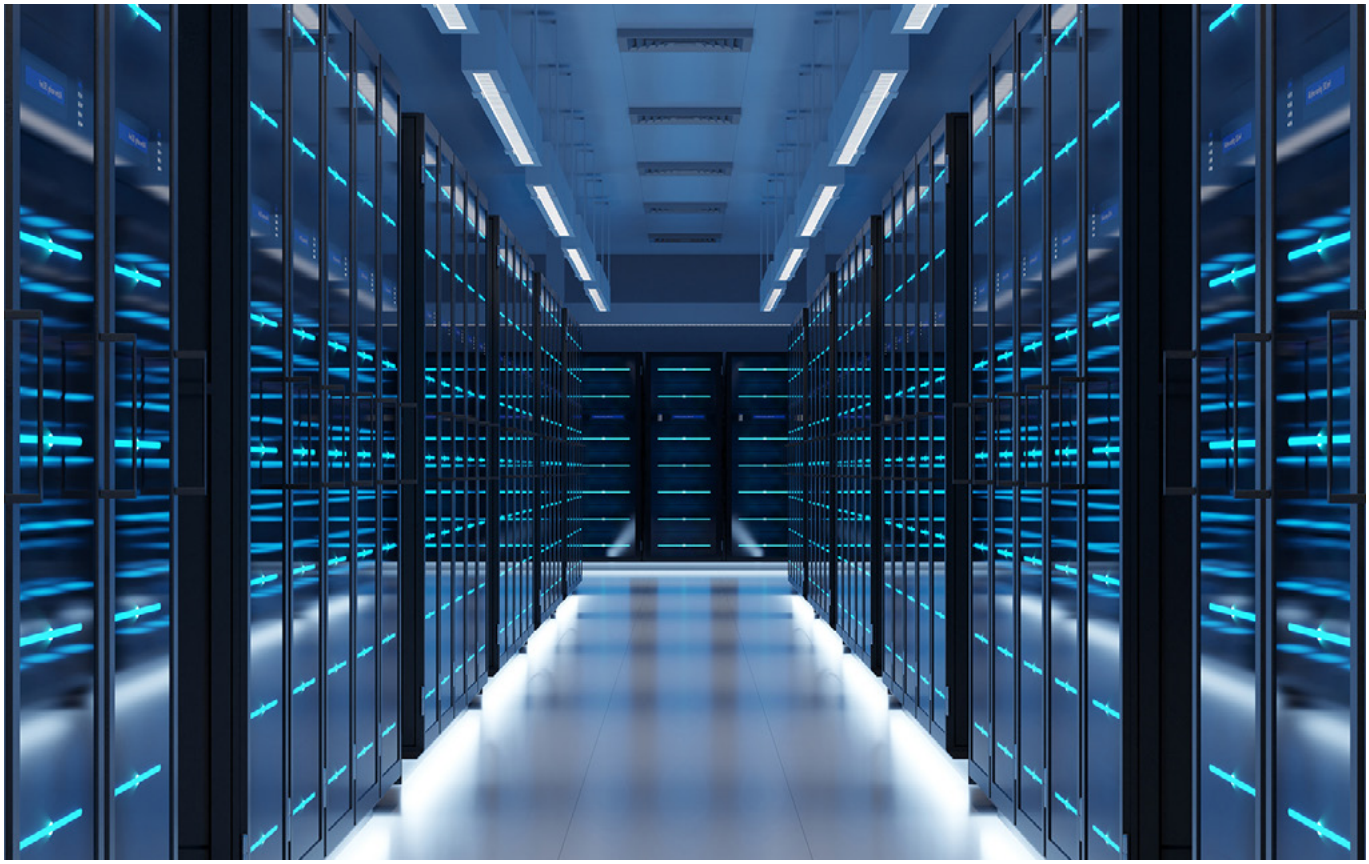


While agentic software engineering adoption is driving faster delivery, it also introduces new risks. According to the DORA 2025 report, every 25% increase in AI adoption correlates with a 7.2% rise in delivery instability. This highlights the need for robust processes and Platform Engineering platforms to manage the pace of change[12].

[12] DORA | State of AI-assisted Software Development 2025

Chapter 3

Laying the foundation for success



Building Platforms where Humans, Agents and Automation converge

Modern software engineering platforms - and the operating models that support them - are now mission-critical for organizations aiming to unlock the full potential of agentic software engineering.

The shift is well underway. Many organizations have already started software engineering transformation towards IDP (Internal Developer Platform) and Platform Engineering, demand is growing, with a clear need for extension to agentic software engineering support. They build, deploy, and operate unified environments with services (mostly self-services) activated by software teams to focus on delivering business software. Meanwhile the platform ensures the highest productivity, quality, security, and compliance under strong governance.

These platforms do more than host code. They orchestrate market leading tools and custom solutions, providing software teams capabilities such as developer workspaces, DevSecOps toolchains, environment provisioning, quality and testing services, lifecycle orchestration, collaboration, and golden paths. These are certified blueprints or 'paved roads' that standardize how different software types are built and operated in context.

Crucially, the Platform Engineering team manages this environment as a product, overseeing build, deployment, operations, and measured adoption. Flexible sourcing models (internal, managed, or hybrid) allow organizations to choose how the platform is operated without compromising control or compliance.

These platforms only succeed when they are designed with users in mind. Adoption depends on strong developer experience, clear discoverability, trust in embedded controls, and intuitive use. When done well, platform engineering goes beyond infrastructure—it enables new ways of working, collaboration, and long-term workforce evolution.

The backbone of agentic software engineering deployment

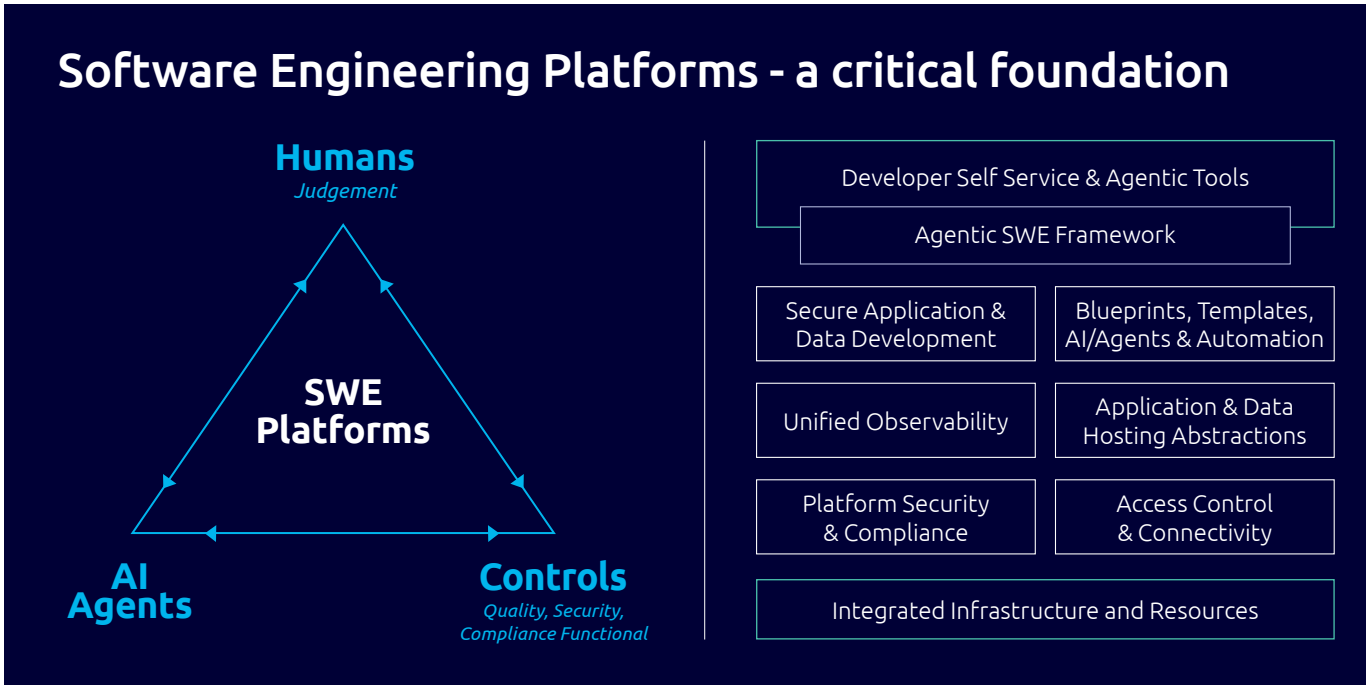
Modern software engineering platforms are the bedrock for successful adoption and scaling of agentic software engineering workflows. They don't just secure productivity and quality, they enforce security, maintain policy, ensure cost control, address sovereignty constraints, and instrument outcomes through a platform level measurement protocol (throughput, quality, compliance, cost, risk). With the right platform, agentic software engineering can be deployed at scale, confidently, with governance built in.

They also create the conditions for workforce evolution. By standardizing workflows, surfacing reusable patterns, and making control points explicit, platforms help organizations train teams more consistently, accelerate progression across maturity levels, and define clearer interfaces between human responsibilities and agent autonomy. This is what turns isolated experimentation into scalable organizational capability.

Software engineering platforms that orchestrate

Our advice is simple but bold. Stop stitching together fragmented toolchains. Build modern platforms that integrate and automate. Move from a toolbox of assets to an orchestrated platform tailored to the context and maturity of the organization. Combine market solutions, targeted accelerators, and robust IDP / Platform Engineering foundations to achieve true scale, security, and adoption. Turn local agentic wins into enterprise grade agentic software engineering value.

By investing in platforms that are adaptive, intelligent, and designed for agentic collaboration, organizations position themselves to not just keep up, but to lead.



Chapter 4

Making agentic software engineering real

Cultivating the capability to evolve

Agentic software engineering is accelerating rapidly, but its full impact will only be understood through timely practice. The ecosystem is evolving too quickly for static assumptions or prescriptive patterns. Real progress comes from doing. From running controlled experiments, learning from variability, and applying those lessons directly to software organizations for delivery.

The first step is to experiment deliberately in real conditions. Early use cases, safe-to-fail prototypes, and focused pilots allow teams to uncover where agentic approaches thrive, where they struggle, and which context is best. Variability in AI outputs becomes a source of insight, rather than a problem to eliminate. It exposes opportunities, constraints, and new ways of working that only emerge through hands-on discovery.

As these insights accumulate, organizations build the understanding required to adapt processes, redefine responsibilities, and evolve engineering practices. This is where learning becomes structural. Patterns solidify, workflows change, and teams begin to design with autonomous agents in mind, rather than bolting them onto existing methods. The feedback loop between experimentation and delivery becomes a catalyst for continuous evolution.

From here, an AI-first mindset emerges. Not as a top-down mandate, but as a natural outcome of accumulated experience. Teams start to assume AI participation by default, framing problems with richer context, designing work as orchestrated workflows, and validating outcomes with new levels of scrutiny. Design becomes a continuous activity. Governance becomes embedded and adaptive. And creativity is amplified by agentic exploration, rather than limited by predefined constraints.

This shift is not about predicting the perfect future team or tooling landscape. It is about cultivating the capability to evolve, anchoring decisions in real delivery evidence, embracing adaptive design, and enabling humans and agents to collaborate fluidly. The organizations that succeed will be those that learn fast, apply faster, and allow their mindset to shift because of what they discover in the field.

Five actions to accelerate your shift

Modern software engineering is not the outcome of scattered AI adoption or the introduction of new tools in isolation. It requires leaders to commit to a coherent, enterprise-wide shift in how software is conceived, built, and evolved.

The following actions provide a pragmatic pathway for organizations seeking to turn ambition into operational change, anchored in clear platforms, repeatable practices, and empowered teams.

1. **Treat your engineering platform as strategic infrastructure**

   A modern software engineering platform is no longer an efficiency enabler. It's the foundation on which agent-driven delivery operates. This means creating a unified environment that integrates governance, automated guardrails, DevSecOps capabilities, testing and observability, self-service environments, and proven delivery pathways. The platform becomes the environment in which humans and AI agents collaborate safely, consistently, and at high velocity.

2. **Target high value, high context areas to demonstrate early impact**

   Meaningful progress begins where both value and context are strong. Areas like documentation generation, code remediation, DevOps automation, reverse engineering, testing, or maintenance. By starting here, organizations can quickly understand where agentic approaches amplify delivery. And where additional context or guardrails are required. Early use cases serve as practical learning engines that inform wider adoption.

3. Reshape teams and reframe roles around orchestration, context setting, and validation

As autonomous and semi autonomous agents take on more execution, human contribution shifts to shaping, directing, and validating outcomes. Teams focus less on manual output and more on providing context, designing workflows, governing behavior, managing exceptions, and ensuring quality — making orchestration, oversight, and structured problem framing core engineering skills.

This shift requires specific role evolution across seniority levels. Junior engineers accelerate their progression through AI-assisted delivery, while senior engineers architects and leads focus more on system design, governance and cross-team orchestration. Clear human-in-the-loop accountability models ensure that ownership for quality, compliance and business outcomes remains well defined, even as execution becomes increasingly automated.

To support this shift, large engineering organizations benefit from being structured into smaller, more focused units. These teams are better able to innovate, adapt, and deliver value faster — not by reducing roles, but by lowering communication overhead and enabling more effective coordination and collaboration.

As execution becomes increasingly automated, competitive advantage shifts toward how organizations redesign teams, rebalance seniority, and define where human judgment, escalation, and governance remain essential.

4. Institutionalize governance and engineering standards for safe scale

As AI becomes embedded across the lifecycle, consistency and clarity matter more than ever. Organizations need standards-driven practices that define architectural patterns, security requirements, quality thresholds, and reusable blueprints. But these practices only become effective when they are reinforced through a culture of software engineering measurement - capturing metrics such as velocity, quality, security and developer experience to verify that AI is driving the intended outcomes. By establishing baselines, instrumenting the platform with clear metrics and KPIs, and automating data collection, teams gain the evidence needed to refine ways of working, optimize tool usage and team compositions, and set well-governed boundaries for both humans and intelligent agents. This not only protects the enterprise but accelerates delivery by reducing ambiguity and rework.

5. Elevate talent, learning and culture as strategic capabilities

Agentic software engineering transforms how work happens and, therefore, where teams need to excel. Organizations must build cultures that encourage exploratory thinking, open knowledge sharing, and continuous learning. Skills such as context engineering, agent orchestration, iterative design, ethical reasoning, and rigorous validation become central to long-term success.

This shift requires a structured approach to capability building, enabling teams to progress from AI-assisted to more autonomous and agentic ways of working. This includes defining clear proficiency levels, embedding learning into delivery, and scaling enablement programs across the organization rather than relying on isolated or ad hoc upskilling.

Equally important are the people sustainability dimensions of this shift. Organizations must rethink career paths, leadership expectations and performance models for smaller, more expert teams. Success depends on creating environments where individuals can continuously evolve their skills, take on higher-value responsibilities and remain engaged as roles shift from execution to orchestration and governance.

The organizations that succeed will treat workforce transformation not as a side effect of AI adoption, but as a deliberate strategy spanning capability building, career pathways, leadership expectations, and long-term people sustainability.

Technology may provide acceleration, but people provide direction, judgment, and the capability to scale safely.

Chapter 5

The Capgemini edge orchestrates ambition into action

C-level executives at software organizations have a crucial three to six month window to define their agentic AI product strategy, as the industry is at an inflection point[13].

Your partner for contextual, scalable, agentic transformation

Navigating the shift to agentic software engineering is not simply a technology decision. It is an enterprise transformation that reshapes delivery models, engineering platforms, governance, and workforce capabilities. Capgemini helps organizations turn that ambition into measurable business outcomes.

We start by establishing a clear view of current maturity across software engineering, platform foundations, governance, and team readiness. This helps identify where agentic approaches can create the greatest value first, and what constraints must be addressed to scale with confidence.

From there, we help clients build the conditions for durable transformation: modern engineering platforms, embedded control points for quality and compliance, and delivery practices that enable effective human–agent collaboration at scale.

Capgemini supports this journey end-to-end through tailored enablement programs, certified blueprints, and scalable adoption paths. Our approach is modular and contextual, not one-size-fits-all. We help clients modernize their platforms, evolve their operating model, and build the skills and roles needed to sustain change over time.

What differentiates Capgemini is our ability to transform the full software engineering system. We do not approach modernization as a tooling exercise alone. We combine platform foundations, workflow redesign, governance, role evolution, and capability building so that technology change becomes lasting workforce and business transformation.

[13] Gartner Predicts 40% of Enterprise Apps Will Feature Task-Specific AI Agents by 2026, Up from Less Than 5% in 2025

How Capgemini turns modern software engineering into execution

We help clients move from ambition to execution through five connected steps:

1. **Assess maturity and readiness.** We establish a clear baseline across platform capabilities, lifecycle practices, governance, and workforce readiness to identify where value can be created fastest and what must be strengthened first.
2. **Prioritize high-value use cases by context.** We focus on the workflows and domains where business value, context quality, and delivery feasibility align, creating early impact while building evidence for broader adoption.
3. **Build the platform and governance backbone.** We design and implement modern engineering platforms with the right integrations, guardrails, control points, and measurement mechanisms so human–agent collaboration can scale securely and consistently.
4. **Redesign roles, teams, and human–agent collaboration.** We help clients evolve team structures, clarify human accountability, and define where orchestration, validation, escalation, and governance remain human-led as agents take on more execution.
5. **Scale enablement, capability building, and adoption.** We embed the change through role-based learning, enablement programs, certified blueprints, and adoption metrics that help clients move beyond pilots to sustained transformation.

Bringing the future into focus…

Software engineering is entering a new era - one shaped not simply by more powerful tools, but by a new partnership between human ingenuity, intelligent agents, and the platforms that bring them together. What lies ahead is more than an evolution in how software is built. It is an opportunity to reimagine how organizations create value, solve problems, and unlock the full potential of their people.

The organizations that will lead in this new landscape will be those that see beyond automation alone. They will understand that the future of software engineering will be defined by their ability to combine speed with judgment, autonomy with accountability, and technological progress with the continuous growth of human capability. In that future, platforms, workflows, and workforce do not evolve separately — they advance together.

For leaders, this is a moment not just to adopt new technologies, but to shape a new generation software organization: more adaptive, more resilient, and more capable of turning intelligence into impact. Those who move now have the opportunity not only to modernize delivery, but to redefine what engineering excellence can look like in the age of intelligent systems.

In the end, the future of software engineering will not be defined by AI adoption alone, but by how effectively organizations combine technological progress with the continuous growth of human capability.

Capgemini helps clients turn that possibility into reality - building the platforms, capabilities, and confidence to shape the next generation software organization.

Authors

Stephane Girard

Chief Technology Officer
Cloud & Custom Applications
(C&CA)

Stéphane Girard is the CTO for Capgemini's Cloud & Custom Applications organization. An accomplished Chief Architect and technology leader, he brings extensive experience across the information technology and services sector. He is highly skilled in designing and delivering enterprise wide platforms and architectures at scale, driving robust, end to end technology solutions.

Inma Casero

Deputy Chief Human Resources Officer and member of Group Executive Committee at Capgemini

Based in Paris, Inma has extensive international experience in human resources, particularly in leading transformation projects and managing culturally diverse teams. Since joining Capgemini in 2012, she has held various HR roles, ultimately reaching her current position as Deputy Chief Human Resources Officer in 2021 and becoming a member of the Group Executive Committee in 2023. Inma also serves as an Independent Director on Broadpeak's Board of Directors. With degrees in Law and Business Administration from Universidad de Castilla La Mancha and Organizational Leadership from Harvard, she is passionate about organizational behavior and regularly contributes to research in this field.

Stuart Williams

Head of Software Engineering
Cloud & Custom Applications UK
(C&CA)

Stuart Williams is Head of Software Engineering at Capgemini UK, where he leads high performing engineering teams and drives modern software delivery practices. An experienced engineering leader and problem solver, he specializes in developer experience, Platform Engineering, and building scalable, enterprise grade solutions that enable teams to deliver software effectively and at pace.

Stuart Ball

Head of Strategy
Cloud & Custom Applications
(C&CA)

Stuart Ball is an Executive Vice President at Capgemini, leveraging extensive leadership experience to help clients drive technology-led transformation and achieve measurable operational improvement. With senior roles across Capgemini and prior consulting positions, he brings deep expertise in delivering solutions that accelerate business performance.

Editorial

Lorna Neville

Global Marketing & Communication Director
Global Cloud & Custom Applications (C&CA)

Lorna Neville is Marketing Director for Cloud & Custom Applications at Capgemini, bringing extensive experience shaping technology-led propositions into clear business value. With senior marketing roles across the Capgemini Group, including leading Customer Experience and multiple industry sectors, she specializes in simplifying complex messaging to support growth and transformation.

About Capgemini

Capgemini is an AI-powered global business and technology transformation partner, delivering tangible business value. We imagine the future of organizations and make it real with AI, technology and people. With our strong heritage of nearly 60 years, we are a responsible and diverse group of over 420,000 team members in more than 50 countries. We deliver end-to-end services and solutions with our deep industry expertise and strong partner ecosystem, leveraging our capabilities across strategy, technology, design, engineering and business operations. The Group reported 2025 global revenues of €22.5 billion.

Make it *real.*
www.capgemini.com

Capgemini