WHITE PAPER

AI Across the Product Development Lifecycle

How enterprise product teams can turn faster coding into faster, safer delivery, from discovery to operations



Varun Vemulapalli

November 2026

Executive Summary

AI has made writing code faster, but in most organizations it has not yet made delivering products faster. Teams that redesign the whole product development lifecycle (PDLC) around AI, rather than adding a coding assistant to the old process, are the ones reporting large and lasting gains.

- **The adoption.** About 90% of technology professionals now use AI at work (DORA, 2025), and Gartner expects 90% of enterprise software engineers to use AI code assistants by 2028 (Gartner, 2025).
- **The paradox.** Controlled studies show task-level gains of 26% to 56%, yet one trial found experienced developers 19% slower with AI, and telemetry from more than 10,000 developers found review time up 91% while company-level delivery metrics stayed flat (Becker et al., 2025; Faros AI, 2025).
- **The opportunity.** In McKinsey's 2026 survey of 334 product and engineering leaders, organizations that redesigned processes before adding AI were more than twice as likely to report gains above 20% (McKinsey & Company, 2026).
- **The risk.** AI-assisted code carries more defects and security flaws than human-only code in several studies, invented package names open a new supply-chain attack, and agents with broad permissions can turn one bad instruction into a production incident.

Recommendations: measure a delivery baseline before scaling tools; fix the flow first (small batches, automated tests, a solid internal platform); make testable specifications the main artifact; scale verification as fast as generation; give agents autonomy by risk tier and prove it in shadow mode; govern under the NIST secure-development and AI risk frameworks, the OWASP LLM and agentic risk lists and, where relevant, the EU AI Act and the EU Digital Operational Resilience Act; and judge success by delivery and customer outcomes, not lines of code.

1. Introduction

The *software development lifecycle* (SDLC) covers how code is designed, written, tested and released. The *product development lifecycle* is wider. It starts earlier, with discovering a customer problem and deciding whether it is worth solving, and it ends later, with operating the product and learning from how people use it. Most of the first wave of enterprise AI went into one narrow part of that cycle: typing code.

That focus made sense. Code is measurable, coding assistants were the first mature tools, and early experiments showed striking speed-ups. But a product reaches customers only as fast as its slowest stage. When code is produced faster than it can be specified, reviewed, tested, secured and released, the work simply queues somewhere else.

This paper looks at AI across the whole PDLC for enterprise product teams, with particular attention to regulated industries such as banking, insurance and healthcare, where an error in production can mean a customer harmed, a regulator involved or both. It is deliberately platform-neutral: the practices apply whichever coding agents, low-code platforms or cloud tools a team uses. It covers what the evidence says, what AI does well and badly at each stage, how roles and controls need to change, and a phased roadmap with exit tests.

2. The Productivity Paradox

Adoption is no longer the question. The 2025 DORA report (Google Cloud's DevOps Research and Assessment program), based on nearly 5,000 technology professionals, found that 90% use AI at work and more than 80% believe it has made them more productive (DORA, 2025). In Stack Overflow's 2025 survey, 84% of developers were using or planning to use AI tools (IT Brief, 2025).

Trust has moved the other way. The share of developers who do not trust the accuracy of AI output rose from 31% in 2024 to 46% in 2025 (Stack Overflow, 2025), and 30% of DORA respondents report little or no trust in AI-generated code (DORA, 2025). The evidence on productivity explains why both trends can be true at once.

Study	Setting	Finding	What it tells product leaders
Peng et al. (2023)	Controlled experiment; one well-defined coding task	Developers with an AI assistant finished 55.8% faster	Large gains on small, clearly specified tasks
Cui et al. (2025)	Randomized trials at Microsoft, Accenture and a Fortune 100 firm; 4,867 developers	26.08% more completed tasks per week; less experienced developers gained most	Real but smaller gains in daily work
Becker et al. (2025)	16 experienced open-source developers, 246 tasks in their own large repositories	Tasks took 19% longer with AI, though developers believed AI had made them about 20% faster	Self-reported speed is not a reliable measure
Faros AI (2025)	Telemetry from over 10,000 developers on 1,255 teams	High-adoption teams completed 21% more tasks and merged 98% more pull requests, but review time rose 91%, pull requests grew 154% and bugs per developer rose 9%; company-level delivery metrics were flat	The bottleneck moves downstream to review and verification
McKinsey & Company (2026)	Survey of 334 product and engineering leaders, May 2026	25% of senior leaders report top acceleration; nearly 30% say team productivity fell; average time saved across PDLC use cases 11.8%, rework reduced 6.2%	Outcomes diverge sharply by operating model, not tool choice

Read together, the studies point to one mechanism. AI makes the *generation* step cheaper, so teams produce more and larger changes. Every downstream step that still runs at human speed, from code review to testing to release approval, absorbs the extra volume. DORA's 2025 data captures the result: AI adoption now has a positive relationship with throughput and product performance, but a negative one with delivery stability (DORA, 2025). DORA describes AI as an amplifier: it magnifies whatever strengths and weaknesses a team already has.

> **Implication.** The constraint in an AI-assisted PDLC is rarely typing. It is clarity about what to build and capacity to verify what was built. Investment should follow the constraint.

3. AI Across the PDLC, Stage by Stage

The table below summarizes where AI is useful today, where human judgment remains essential, and the control each stage needs. It reflects the evidence in this paper and the author's delivery experience; it is a starting point to adapt, not a standard.

Stage	What AI does well today	What stays human	Control to add
1. Discover	Synthesizes interviews, support tickets, reviews and usage data; clusters pain points; drafts opportunity summaries	Choosing which problem matters; talking to real customers	Trace every insight to its source data; watch for invented quotes
2. Define	Drafts requirements, user stories and acceptance criteria; flags gaps, conflicts and missing edge cases	Product intent, priorities, regulatory interpretation	Testable specifications with clear definitions of done, reviewed before any build starts
3. Design	Generates UX variants, prototypes, data models and architecture options; analyzes legacy code and documents	Architecture decisions, trade-offs, accessibility and brand judgment	Architecture decision records; threat modeling for new data flows
4. Build	Completes and writes code, refactors, migrates frameworks; agents implement whole tasks from a spec	Owning the change; deciding what not to build	Small, scoped tasks; least-privilege agent permissions; approved dependency lists
5. Verify	Generates unit and regression tests, test data and security scans; reviews pull requests against the spec	Judging whether the tests prove the right thing; risk acceptance	Independent verification (a different model or tool from the one that wrote the code); coverage of acceptance criteria
6. Release	Drafts release notes and change records; summarizes risk; checks policy before deployment	Go or no-go for high-risk changes	Risk-tiered approvals; small batches; automated rollback
7. Operate and learn	Triages incidents, summarizes logs, proposes fixes; analyzes feedback for the next cycle	Customer communication; incident command	Human approval for production fixes; post-incident review of any AI-proposed change

3.1 Discover and define: where the leverage starts

Product management work benefits from AI as much as coding does. In a McKinsey study of product managers, generative AI tools improved PM productivity by 40% and shortened time to market by 5% over a six-month PDLC, largely by speeding up research synthesis and early drafting (Gnanasambandam et al., 2024). The same study found that output quality depended heavily on PM experience, which is a reminder that AI accelerates judgment but does not supply it.

The most important change at this stage is the artifact itself. A requirement written for a human developer can rely on shared context. A requirement handed to a coding agent cannot. This is why *spec-driven development* has gained ground: GitHub open-sourced its Spec Kit in September 2025 to make a written specification, technical plan and task list the source that coding agents work from, instead of ad hoc prompts (GitHub, 2025). McKinsey's leading organizations make the same move, raising requirements to testable specifications that an agent, an engineer, a test suite and a reviewer can each act on without follow-up questions (McKinsey & Company, 2026).

3.2 Build and verify: generation is cheap, verification is not

Coding agents can now take a task from a specification to a pull request. The quality evidence argues for strong verification. In an analysis of 470 open-source pull requests, AI-co-authored changes contained about 1.7 times as many issues as human-only ones, with logic and correctness issues 75% more common (CodeRabbit, 2025). Veracode tested more than 100 models on 80 coding tasks and found security flaws in 45% of cases, with no improvement in newer models (Veracode, 2025). A USENIX Security study of 576,000 generated code samples found that at least 5.2% of packages suggested by commercial models, and 21.7% by open-source models, did not exist (Spracklen et al., 2025). Attackers can register those invented names, a technique now called slopsquatting.

AI also helps on the verification side. In the 2025–26 World Quality Report, which surveyed more than 2,000 senior executives, nearly 90% of organizations were pursuing generative AI in quality engineering, and they reported an average productivity gain of 19%, though only 15% had scaled it across the enterprise (Capgemini et al., 2025). The practical rule: verification capacity, much of it automated, must grow at least as fast as generation capacity, and the checker should be independent of the generator.

3.3 Release and operate: protect stability

Because DORA links AI adoption to lower delivery stability, release practices matter more, not less. Small batches, automated tests in the pipeline, feature flags and fast rollback limit the damage from any single faulty change. DORA also found that 90% of organizations have adopted at least one internal platform and that high-quality platforms correlate with an organization's ability to get value from AI (DORA, 2025). A good platform gives every team, and every agent, the same safe path to production.

4. The Operating Model Shift

McKinsey's 2026 research found that technology was not what separated the leaders. The top accelerators shared four traits: they redesigned workflows end to end, changed roles and team structures, built verification and measurement into the delivery flow, and invested heavily in hands-on change management (McKinsey & Company, 2026).

- **Smaller teams with broader roles.** Among top accelerators, 79% reported smaller squads, with the median falling from about ten people to seven. McKinsey describes two emerging archetypes: *definers*, who combine product and design, and *builders*, who combine development, testing and operations.
- **A shared AI operations layer.** Leaders centralize identity, permissions, tool access, evaluation, logging, escalation and cost visibility for agents rather than letting each team improvise. McKinsey notes that spending on tokens, compute and this layer may reach 20% of existing labor costs, so it needs a business case of its own.
- **Change management, not licenses.** A global bank in the study saw only 10–15% gains from a copilot rollout; a structured change program with embedded AI engineers later produced efficiency gains of 40–80%. At the software company Sonar, teams that embedded agents across the cycle cut pull-request cycle time by a factor of 3.4 and raised pull-request throughput by a factor of 2.2.

Two cautions apply. Survey and case-study figures come from organizations that chose to report them, so they show what is possible, not what is typical. And the gains came from redesigning work, which takes leadership time that a tool rollout does not.

5. Risk, Security and Governance

5.1 The main risks

- **Insecure and incorrect code.** Higher defect and vulnerability rates in AI-assisted changes (CodeRabbit, 2025; Veracode, 2025), made worse when larger pull requests receive lighter review.
- **Supply-chain attacks.** Hallucinated or unvetted dependencies that attackers can publish under the invented name (Spracklen et al., 2025).
- **Prompt injection.** An issue, document, web page or code comment can carry hidden instructions to an agent that reads it. Prompt injection remains the first entry in the 2026 OWASP Top 10 for LLM Applications (OWASP GenAI Security Project, 2026).
- **Excessive agency.** An agent with more tools, permissions or autonomy than its task needs can turn a small error into a production incident or data leak. Excessive agency rose to third place in OWASP's 2026 list, and OWASP now publishes a separate Top 10 for Agentic Applications covering risks such as agent goal hijack and tool misuse (OWASP GenAI Security Project, 2025, 2026).
- **Data and IP leakage.** Source code, customer data or secrets sent to tools without approved data-handling terms.
- **Over-reliance and skill erosion.** Reviewers who approve plausible code they do not fully understand, and junior staff who never build the skills reviewing requires.
- **Runaway cost.** Agent usage that scales with activity rather than value.

5.2 Frameworks that apply

Framework	Who it applies to	What it means for an AI-assisted PDLC
NIST Secure Software Development Framework (SP 800-218) and its generative AI profile (SP 800-218A)	Voluntary; widely used as a baseline, including by software suppliers to the US government	Secure development practices for all code, plus AI-specific practices for teams that build or use AI models across the development lifecycle (Booth et al., 2024)
NIST AI Risk Management Framework	Voluntary, any organization	Four functions, govern, map, measure and manage, for the AI tools a team uses and the AI features it ships (National Institute of Standards and Technology, 2023)
OWASP Top 10 for LLM Applications (2026) and Top 10 for Agentic Applications	Industry guidance	Threat checklists for AI features and agents: prompt injection, sensitive information disclosure and excessive agency lead the 2026 LLM list; the agentic list adds risks such as agent goal hijack, tool misuse and identity and privilege abuse
EU AI Act	Providers and deployers of AI systems in the EU	Applies mainly to AI features in the product, not to coding assistants; transparency duties apply from 2 August 2026, and high-risk obligations from 2 December 2027 (Annex III) and 2 August 2028 (Annex I) (Future of Privacy Forum, 2026)
EU Digital Operational Resilience Act (EU DORA)	EU banks, insurers, investment firms and other financial entities, and their ICT providers	Applies since 17 January 2025; ICT risk management, incident reporting, resilience testing and third-party risk rules cover the AI tools and vendors in the delivery chain (Mayer Brown, 2025)

The EU AI Act timeline reflects the Digital Omnibus on AI approved by the Council of the EU on 29 June 2026 (Future of Privacy Forum, 2026; Secure Privacy, 2026). Note that EU DORA, the regulation, is unrelated to DORA, the research program cited elsewhere in this paper.

5.3 Controls to build into the delivery pipeline

1. Classify changes by risk tier. Low-risk changes, such as documentation, tests and internal tooling, can flow with automated checks; customer-facing, regulated, security-sensitive or architectural changes need stronger evidence and named human approval (McKinsey & Company, 2026).
2. Give each agent a scoped mandate: the repositories, tools and environments it may touch, with no production credentials by default.
3. Treat every external input an agent reads, including tickets, documents and web content, as untrusted, and keep secrets out of agent context.
4. Check every new dependency against an approved registry before it can be installed, and run software composition analysis in the pipeline.
5. Keep pull requests small and require that each one traces to a specification and its acceptance criteria.
6. Use independent verification: automated tests, static analysis and security scanning that do not rely on the model that wrote the code.
7. Record which changes were AI-assisted, which agent and model produced them, and who approved them, so defects and incidents can be traced back.
8. Maintain an inventory of AI tools and agents with owners, data-handling terms and risk tiers, mapped to the NIST functions and, for EU financial entities, to EU DORA third-party registers.

6. Measuring What Matters

The most common measurement mistake is to track activity: licenses issued, suggestions accepted, lines generated. These numbers rise with any rollout and say nothing about value. In McKinsey's survey, 86% of top accelerators tracked outcome metrics, while low accelerators relied more on tool adoption figures (McKinsey & Company, 2026). A balanced scorecard has three layers.

Layer	Metrics	Why it matters
Delivery	DORA's four keys: deployment frequency, lead time for changes, change failure rate and time to restore service	Shows whether faster coding became faster, stable delivery
Flow and quality	Review time, pull-request size, rework rate, escaped defects, security findings per release, share of changes traced to a specification	Detects the downstream bottleneck and quality drift early
Outcome and cost	Time from idea to customer value, customer and employee experience, AI cost per delivered change	Connects AI spend to business results

Measure a baseline for at least one quarter before scaling, compare teams that adopt with similar teams that have not yet adopted, and report the bad news alongside the good. A rise in throughput with a rise in change failure rate is not a win.

7. Implementation Roadmap

The roadmap below is the author's framework, ordered so that each phase removes the constraint the next one would otherwise hit. Each phase ends with an exit test.

1. **Baseline and guardrails.** Name an accountable leader for AI in delivery. Approve tools against data-handling and security requirements, publish an acceptable-use policy, and measure the delivery baseline. Exit test: one quarter of baseline metrics and an approved tool list.
2. **Fix the flow.** Before scaling generation, invest where work will queue: automated test coverage, a reliable CI/CD pipeline, a shared internal platform, small-batch habits and review capacity. Exit test: review time and change failure rate stable or improving while pilot teams increase output.
3. **Specifications first.** Adopt a spec template with acceptance criteria, definitions of done and risk tier, and pilot spec-driven development on one value stream from discovery to release. Exit test: most pilot changes trace to a reviewed specification, and rework falls.
4. **Agents by risk tier.** Introduce coding and verification agents for low-risk work first. Run them in shadow mode on higher tiers, where the agent proposes and people still decide, and widen autonomy one tier at a time as evidence builds, using the same evidence-led method the author has described for straight-through processing. Exit test: defined evidence thresholds met per tier, with every autonomy change approved and recorded.
5. **Scale and reshape roles.** Redesign team structures around definer and builder roles, fund the shared AI operations layer, and run adoption as a change program with coaching and protected learning time. Exit test: delivery and outcome metrics improve across value streams, not just in pilots.

8. Workforce Implications

- **Juniors gain most, at first.** Less experienced developers adopted AI more and gained more in the large field trials (Cui et al., 2025), while experienced developers on familiar code saw no gain or a slowdown (Becker et al., 2025). Teams should plan for both.
- **Review becomes a core skill.** When generation is cheap, reading, testing and challenging code and specifications is where quality is won. Organizations should train for it and protect time for it.
- **Guard the apprenticeship.** If juniors never write or debug code themselves, the pipeline of future reviewers dries up. Deliberate practice, pairing and rotation through verification work help.
- **Everyone upskills.** Gartner expects about four in five software engineers to need upskilling by 2027 as generative AI reshapes their roles (HR Dive, 2024). In McKinsey's survey, lack of time to learn was a top barrier for about 38% of respondents, and embedded coaching was about 30% more common among top accelerators (McKinsey & Company, 2026).
- **Product roles change too.** Product managers spend less time drafting and more time on judgment, customer contact and specification quality. Top-accelerator PMs reported a 24% reduction in execution time (McKinsey & Company, 2026).

9. Conclusion

AI has already changed how code is written. Whether it changes how products are delivered depends on choices that have little to do with the tools: how clearly teams define what to build, how much verification capacity they build alongside generation, how they grant autonomy to agents, and what they choose to measure.

The organizations pulling ahead treat AI as a reason to redesign the whole lifecycle. They start from a measured baseline, fix the flow before adding volume, make testable specifications the center of the work, scale verification with generation, widen agent autonomy only as evidence builds, and judge success by stable delivery and customer outcomes. In regulated industries that discipline is not optional, and it is also what turns faster coding into faster, safer products.

References

Becker, J., Rush, N., Barnes, B., & Rein, D. (2025). *Measuring the impact of early-2025 AI on experienced open-source developer productivity*. METR. https://arxiv.org/abs/2507.09089

Booth, H., Souppaya, M., Vassilev, A., Ogata, M., Stanley, M., & Scarfone, K. (2024). *Secure software development practices for generative AI and dual-use foundation models: An SSDF community profile* (NIST SP 800-218A). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-218A

Capgemini, Sogeti, & OpenText. (2025, November 13). *World Quality Report 2025: AI adoption surges in quality engineering, but enterprise-level scaling remains elusive* [Press release]. https://www.capgemini.com/us-en/news/press-releases/world-quality-report-2025-ai-adoption-surges-in-quality-engineering-but-enterprise-level-scaling-remains-elusive/

CodeRabbit. (2025, December 17). *Our new report: AI code creates 1.7x more problems*. https://www.coderabbit.ai/blog/state-of-ai-vs-human-code-generation-report

Cui, Z., Demirer, M., Jaffe, S., Musolff, L., Peng, S., & Salz, T. (2025). The effects of generative AI on high-skilled work: Evidence from three field experiments with software developers. *Management Science*. https://doi.org/10.1287/mnsc.2025.00535

DORA. (2025). *State of AI-assisted software development 2025*. Google Cloud. https://dora.dev/dora-report-2025/

Faros AI. (2025). *The AI productivity paradox report 2025*. https://www.faros.ai/blog/ai-productivity-paradox-report-2025

Future of Privacy Forum. (2026). *The AI Act implementation timeline: What changes under the AI Omnibus?* https://fpf.org/blog/the-ai-act-implementation-timeline-what-changes-under-the-ai-omnibus

Gartner. (2025, July 1). *Gartner identifies the top strategic trends in software engineering for 2025 and beyond* [Press release]. https://www.gartner.com/en/newsroom/press-releases/2025-07-01-gartner-identifies-the-top-strategic-trends-in-software-engineering-for-2025-and-beyond

GitHub. (2025, September 2). *Spec-driven development with AI: Get started with a new open source toolkit*. https://github.blog/news-insights/product-news/spec-driven-development-with-ai-get-started-with-a-new-open-source-toolkit/

Gnanasambandam, C., Harrysson, M., Singh, R., & Yarlagadda, C. (2024, May 31). *How generative AI could accelerate software product time to market*. https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/how-generative-ai-could-accelerate-software-product-time-to-market. McKinsey & Company.

HR Dive. (2024, October 4). *Generative AI will spark mass upskilling of software engineers, Gartner says*. https://www.hrdive.com/news/software-engineers-upskilling-generative-AI-Gartner/728988/

IT Brief. (2025, July 30). *Developers adopt AI tools but trust issues persist, survey finds*. https://itbrief.com.au/story/developers-adopt-ai-tools-but-trust-issues-persist-survey-finds

Mayer Brown. (2025, January). *Cybersecurity in the financial sector: EU's Digital Operational Resilience Act takes effect*. https://www.mayerbrown.com/en/insights/publications/2025/01/cybersecurity-in-the-financial-sector-eus-digital-operational-resilience-act-takes-effect

McKinsey & Company. (2026, August 21). *Beyond the copilot: Scaling the agentic product development life cycle*. https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/beyond-the-copilot-scaling-the-agentic-product-development-life-cycle

National Institute of Standards and Technology. (2023). *Artificial intelligence risk management framework (AI RMF 1.0)* (NIST AI 100-1). https://doi.org/10.6028/NIST.AI.100-1

OWASP GenAI Security Project. (2025, December 9). *OWASP Top 10 for agentic applications 2026*. https://genai.owasp.org/2025/12/09/owasp-top-10-for-agentic-applications-the-benchmark-for-agentic-security-in-the-age-of-autonomous-ai/

OWASP GenAI Security Project. (2026). *OWASP GenAI LLM top 10 2026*. https://genai.owasp.org/resource/owasp-genai-llm-top-10-2026/

Peng, S., Kalliamvakou, E., Cihon, P., & Demirer, M. (2023). *The impact of AI on developer productivity: Evidence from GitHub Copilot*. https://arxiv.org/abs/2302.06590

Secure Privacy. (2026). *EU AI Act Digital Omnibus: The new high-risk AI deadlines after Council approval*. https://secureprivacy.ai/blog/eu-ai-act-digital-omnibus-the-new-high-risk-ai-deadlines-after-council-approval

Spracklen, J., Wijewickrama, R., Sakib, A. H. M. N., Maiti, A., Viswanath, B., & Jadliwala, M. (2025). We have a package for you! A comprehensive analysis of package hallucinations by code generating LLMs. In *Proceedings of the 34th USENIX Security Symposium* (pp. 3687–3706). https://www.usenix.org/conference/usenixsecurity25/presentation/spracklen

Stack Overflow. (2025). *Stack Overflow's 2025 Developer Survey reveals trust in AI at an all time low* [Press release]. https://stackoverflow.co/company/press/archive/stack-overflow-2025-developer-survey/

Veracode. (2025, July 30). *AI-generated code poses major security risks in nearly half of all development tasks, Veracode research reveals* [Press release]. https://www.veracode.com/press-release/ai-generated-code-poses-major-security-risks-in-nearly-half-of-all-development-tasks-veracode-research-reveals/

About this paper

Author: Varun Vemulapalli. **Published:** November 2026.