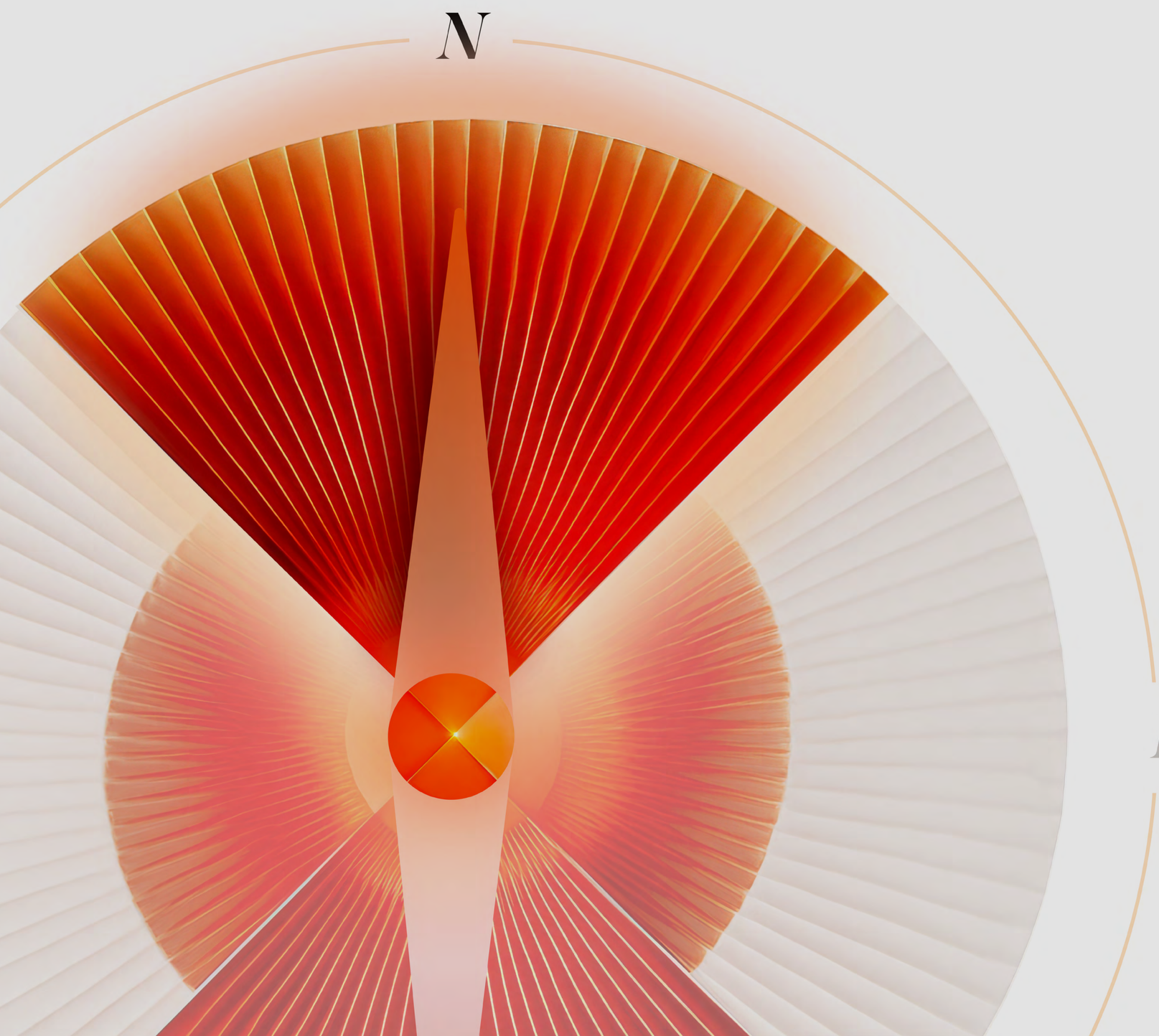
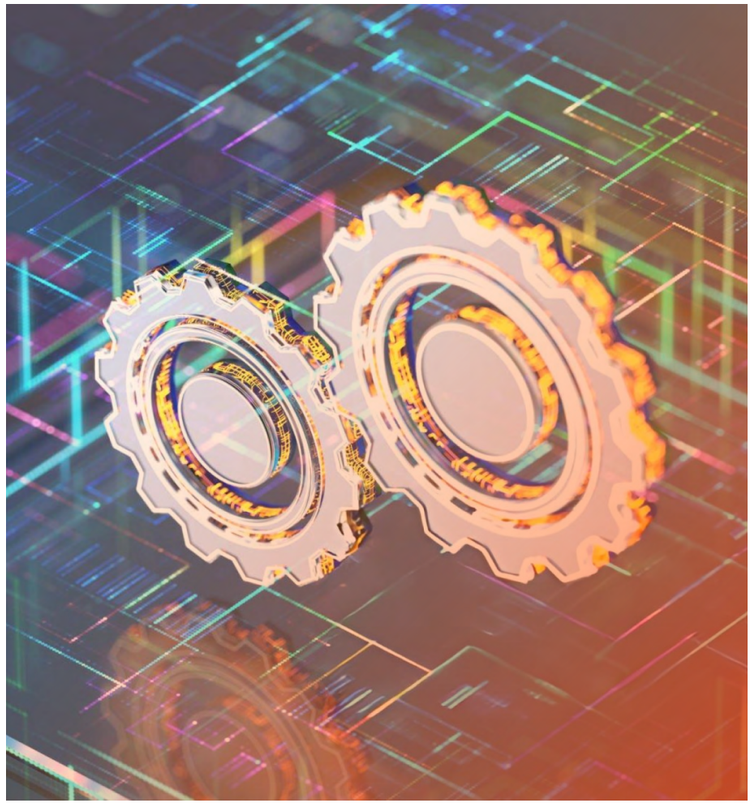
sopra steria next



No Train, *No Gain*

From typing code to curating working software

CIO Compass Series



A new *engineering reality*

Generative AI hasn't just "made coding faster." It has shifted software to **end-to-end delivery:** clear intent, test evidence, safe integration, and reliable runtime behavior. Like the CAPEX→OPEX shift, this is first a **governance change:** manage outcomes (not activity), contain variability, and make economics visible at the **user-story** level.

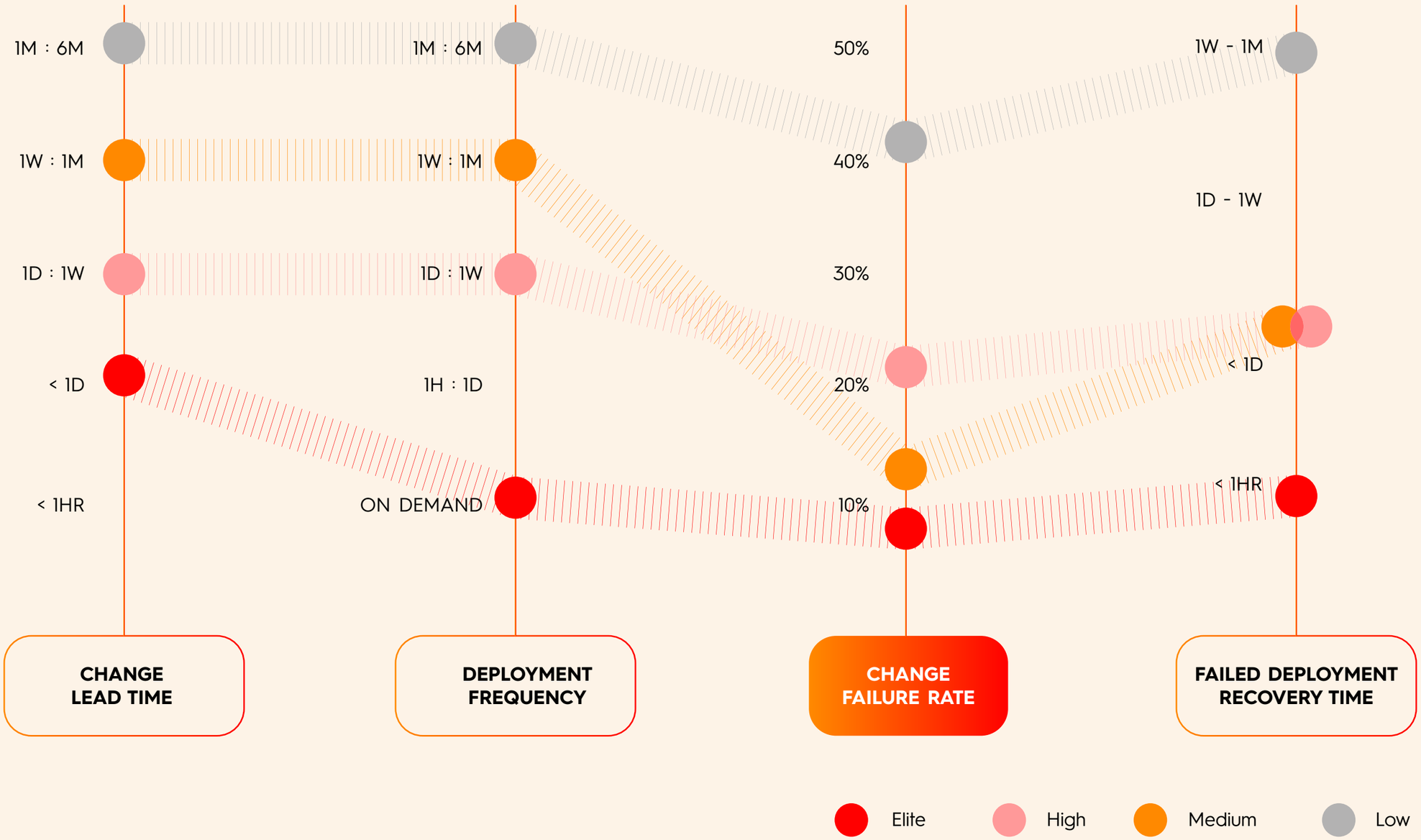
Why *this matters now*

Boards no longer reward pilots; they want **repeatable, auditable delivery** with controlled model spend, compliant data flows, and adoption measured beyond demos. The question isn't *which tool?* but **how to run software as a disciplined system** where AI is present in every step and value is visible. According to industry benchmarks, (Elite) high-performing teams achieve Change Failure Rates below 15% (versus approximately 31-45% for medium-performing teams[1]), illustrating the difference between mere pilots and truly reliable production delivery[2].

> **High-performing teams achieve Change Failure Rates below *15%.***

1 What Are DORA Metrics? A Guide for DevOps Teams | New Relic
2 4 Key DevOps Metrics to Know | Atlassian

Software delivery performance levels



Based on *DORA, Accelerate State of DevOps*, 2024, Google Cloud.

Modern assistants now operate at **repository scale.** Anthropic's Claude Sonnet 4 introduced a **1-million-token** context window–enough to ingest large codebases and reason across them for multi-file refactors and cross-module fixes. Google's Gemini 1.5 Pro opened a **1-million-token** window and documents whole-repo analysis in developer materials.

Beyond "reading," agents increasingly **do the work:** GitHub Copilot's coding agent can take an issue, spin up a VM, clone the repo, run tests, and submit a PR, delegated, repo-level tasks rather than autocomplete. Early enterprise cases also show **hours of sustained autonomous coding** under guardrails (e.g., Rakuten's 7-hour continuous refactor and major cycle-time cuts).

"Vibe coding": promise and policy

These capabilities fuel a cultural shift often called **"vibe coding":** describe intent conversationally and let an agent generate or modify the software, judging the result by behavior more than the code. Used well, it **lowers the barrier to exploration:** faster sketches, broader participation. Used recklessly, it **bypasses engineering discipline,** creating messy, insecure, or unmaintainable outputs. Enterprise stance: allow vibe-style sketches only in **sandboxes** with exit criteria (specs + tests + runtime signals), pair non-experts with orchestrator developers, and enforce a light **"vibe → viable"**checklist.

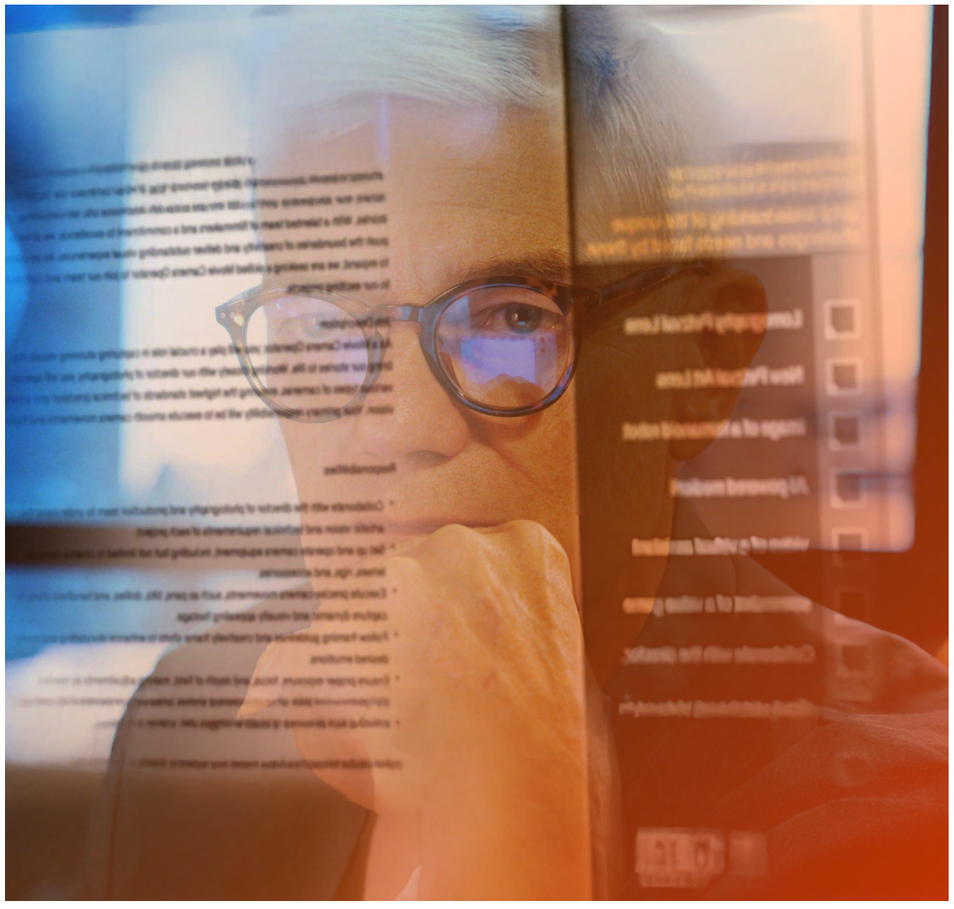
Don't optimise 7.5% of the problem – *unlock the Software Development Life Cycle (SDLC)*

Developers spend roughly a third of their time typing, and "development" is only a fraction of project effort. Optimising **code generation alone** barely moves outcomes. In practice, teams deploying AI across the SDLC report up to 31,8% reduction in Pull Request review times in real-world settings[3]. Real gains come from spreading AI across the **entire SDLC** so small, reliable improvements **compound:** tighter discovery and scoping; specifications with complete acceptance criteria; faster, better-argued architectures; safer reviews; earlier, stronger tests; augmented security and compliance; assisted pipelines and troubleshooting; documentation generated from code and stories; and copilots that accelerate incident response. **Stability and throughput rise when AI touches every stage, not just coding.** Recent studies report 7-10% cost savings, and 2-10 hours of expert time saved per week when AI is embedded across the SDLC[4].

3 [2509.19708] Intuition to Evidence: Measuring AI's True Impact on Developer Productivity
4 AI in software development: The complete guide to tools, productivity & real ROI | LinearB Blog

Why govern *by user story* (not tasks)

AI scrambles activity metrics. Keystrokes and hours no longer map to value, while **risk and rework** migrate to specification, testing, and integration. The **user story** is the smallest unit that captures intent, evidence, cost, and runtime behavior–and the only place you can see whether AI helped or just moved work downstream. If value is delivered as stories, **govern where value lives.**



Govern *by user story*

AI doesn't cut cost by decree; it **reshapes unit economics.** Two leadership moves make it real:

- **Expose costs and value per story.** Treat token/GPU like any input and attribute it to the story that consumed it; visibility kills waste fast.
- **Reinvest speed in design and quality.** If time-savings turn into more scope, you create a mirage. The compound benefit appears when faster generation funds better specs, earlier tests, and cleaner architectures.

Make these visible per product and per sprint. Fragile patterns and wasteful prompts don't survive–and the real gains (fewer loops, earlier tests, safer releases) become measurable and repeatable.

Five non-negotiables signals (KPIs):

- **Lead time to user story** (idea to production)
- **Escaped defects** (what still hits prod)
- **Rework after AI** (post-merge effort caused by AI output)
- **Useful coverage** (tests on risk paths)
- **Unit cost per story** (people, environments, defects, AI spend)

The *productivity shock* (without the math)

Keep the steering light: the five signals above are enough to run–and to avoid metric gaming.

Roles & skills *– what actually changes*

This isn't the end of developers; it's the end of the **single-skill code typist.**

- From **Developers to Orchestrators** of intent & integration (constrain prompts, assemble safely, explain diffs).
- From **Tech leads / architects to Guardians of vision** (guardrails, curated prompts/patterns, trade-offs).
- From **QA/SET** to **Quality advocates** (test strategy, validation of AI-generated suites, coverage on risk paths).
- From **PM/PO** shift time to specification quality, validation, and adoption–**generation speed ≠ delivery speed.**

Careers hinge on five literacies: **algorithmic thinking, testing strategy, pragmatic architecture, AI-FinOps, ethics/IP hygiene.**

The talent shock: *don't lose a junior generation*

AI eats the entry layer first (boilerplate, glue, basic tests)–the work juniors used to learn on.
If nothing changes, seniors absorb more; juniors become prompt operators; capability erodes and attrition rises. Treat apprenticeship as a system, not goodwill: short dojos on live code, predictable pairing each week, and a three-step rotation that moves people from curate AI output (explain diffs, harden prompts), to author specs & tests (acceptance criteria, property tests, synthetic data), to design interfaces (APIs, failure modes, basic security).

Keep room in every sprint for learning stories–tighten an API contract, add observability, address a non-functional, or run a safe refactor. Review progress rather than scores:

- **Prompt hygiene:** the prompt states intent, constraints, examples, and how you'll check the result.
- **Evidence quality:** tests include negatives/boundaries and have already caught something real.
- **Interface thinking:** the contract documents errors, limits, versions, and is easy to read.
- **Runtime literacy:** the change is visible in logs/traces and tied to a basic reliability goal.

AI in the SDLC: Turning Speed into Software performance

Our working group illustrates how AI can transform the Software Development Life Cycle (SDLC) into a strategic driver of quality, security, and business value.

At Sopra Steria, software productivity is at the core of our businesses – from Tech Advisory to Systems Integration. Our software factory enables faster, safer, and large-scale delivery through automation and continuous integration.

AI now amplifies this model. It turns software engineering into a true catalyst for innovation and performance. More than 90% of developers already use AI tools daily; the next challenge is scaling adoption responsibly.

Across the industry, a joint working group reported 7–10% cost savings, 2–10 expert hours saved per week, and up to 40% fewer bugs.

Sopra Steria helps clients embed AI across every stage of the SDLC: from code generation and automated testing to documentation, through tailored training and governance frameworks.

Well-governed AI doesn't just accelerate development; it redefines software performance.

Operating model: the four no-regret moves for a lightweight, industrial and sovereign delivery system

01

Product teams with evidence gates. A story is ready only when intent, examples, and allowed data are clear; it is done only when tests are validated, prompts traceable, and security checks pass. Teams own these five signals: their shared compass for balancing speed, quality, and control, so that acceleration doesn't turn into new technical debt.

02

A small AI Platform team (the paved road). One set of CI/CD templates, one governed model gateway, default observability, automatic security checks, and AI-cost tagging per story–**simplify** and keep everyone on a fast, safe path.

03

A PromptOps Guild (not a department). A short weekly circle that versions prompts like code, curates patterns, tracks impact (less rework, better coverage), and retires bad recipes–preventing "shadow prompts."

04

A lightweight Model & Data Council. A brief, bi-weekly gate for **safety, cost, and sovereignty:** model allow-lists by use case, rules for sensitive data, and time-boxed exceptions. Everything else stays with product teams.

Together, these moves deliver speed, proof, and control without bureaucracy.

Hidden traps

- **Plug-and-prompt without guardrails,** runaway spend, variability, IP exposure.
- **Obsolete productivity metrics,** teams type faster but don't ship better.
- **Shadow AI** (unmanaged tools/prompts) hides risk and cost until it's too late.
- **Confusing generation speed** with delivery speed undermines adoption and credibility.

Call to action

- **Train now** as skills move faster than org charts.
- **Change the unit of measure** and govern by user story with visible economics.
- **Industrialise guardrails** so speed doesn't become new debt.
- **Reinvest** every gain in design and quality, where compounding advantage lives.

Conclusion

If AI touches only your code, you'll get faster keystrokes and the same old problems. When it touches your **whole SDLC,** measured **by user story,** you get **velocity you can trust.** That's the difference between impressive demos and durable advantage.

Contacts

Julien MASSON
Group Head of Tech Advisory
→ julien.masson2@soprasterianext.com

Contributors of our European Network

Olivier BROUSSEAU
Architecture & Tech
→ olivier.brousseau@soprasterianext.com

Isabelle PONS
Data & AI
→ isabelle.pons@soprasterianext.com

Valérie THIBLET
Data & AI
→ valerie.thiblet@soprasterianext.com

Axelle ROESCH
CIO Advisory
→ axelle.roesch@soprasterianext.com

Thomas OTTO
CIO Advisory
→ thomas.otto@soprasteria.com

Philipp MAIER
CIO Advisory
→ philipp.maier@soprasteria.com

Simon KAYE
Architecture & Tech
→ simon.kaye@soprasteria.com

Ray BAKER
Architecture & Tech
→ ray.baker@soprasteria.com

Andy WHITEHURST
Architecture & Tech
→ andy.whitehurst@soprasteria.com

Eirik HASLESTAD
CIO Advisory
→ eirik.haslestad@soprasteria.com

Jahn-Fredrik SJOVIK
CIO Advisory
→ jahn-fredrik.sjovik@soprasteria.com

Tor NESET
Architecture & Tech
→ tor.neset@soprasteria.com

Marius SANDBU
Data & AI
→ marius.sandbu@soprasteria.com

Robert HEXSPOOR
Architecture & Tech
→ robert.hexspoor@soprasteria.com

Hugo HERMSEN
CIO Advisory
→ hugo.hermsen@soprasteria.com

Juan Antonio MARTINEZ SANCHEZ
Architecture & Tech
→ juanantonio.martinezsanchez@soprasterianext.com

Mauro PALMARINI
CIO Advisory
→ mauro.palmarini@soprasterianext.com

Conception: Spintank, photos credits: GettyImages ©