Escape AI pilot mode

Three ways engineering teams turn AI experiments into measurable delivery improvements.

Why do most AI pilots fail?

If it's not AI, what is the real bottleneck?

What are engineering teams missing about delivery?

nearform.com

Nearform

Why does progress stall after pilots?

Is AI solving the wrong problems?

Why AI Initiatives get stuck in pilot mode

What actually changes delivery?

What we're seeing_

AI experimentation is now widespread across enterprise engineering teams. AI tools generate code, agents assist with testing, and automation workflows support the entire delivery lifecycle.

Yet despite heavy investment, many organisations still struggle to show measurable improvements in delivery performance. The issue is rarely the technology, but how it is applied. In practice, this shows up in a few consistent ways:

- Most organisations introduce these tools without changing the systems responsible for building, testing, and running software, particularly in brownfield environments where legacy code and accumulated technical debt make change difficult

---

- Pilots, typically small-scale AI experiments outside core delivery workflows and production systems, create isolated gains but fail to enhance how software is produced or released

---

- Progress then stalls when integration, security, or governance concerns surface late

---

- As AI begins to interact with production systems and sensitive data, existing architectures are exposed as unprepared to manage the risks

The result_

Many organisations struggle to move beyond the early stages of AI adoption. There is plenty of activity but little measurable improvement in delivery performance.

Escaping pilot mode requires a different approach. Instead of experimenting with AI in isolation, teams must apply it where it directly changes delivery outcomes.

Three approaches have proven reliable starting points.

THREE RELIABLE STARTING POINTS

01 Start where teams lose the most time

02 Experienced engineers unlock AI's real value

03 Use AI to solve the problems that have stalled for years

01_ Start where teams lose the most time

Every software organisation carries a layer of repetitive tasks that quietly drain developer capacity. These jobs are necessary to maintain a healthy codebase, but over time they accumulate and slow delivery.

These tasks are also well-suited to AI. Many follow clear patterns and can be validated through testing and review, making them relatively safe to automate.

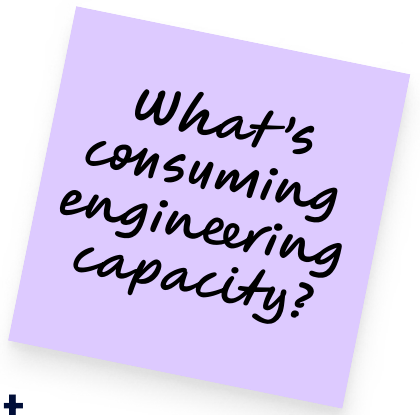
Examples include:

- Generating unit and integration tests
- Refactoring legacy code or modernising frameworks
- Producing documentation for complex codebases
- Upgrading dependencies and patching security vulnerabilities
- Analysing logs and identifying likely causes of bugs

Applying AI to this development ‘toil’ frees senior developers from hours of routine maintenance. That time can instead be spent on architecture, product features, and system reliability.

The result is faster delivery and early confidence in applying AI to production workflows.

02_ Experienced engineers unlock AI's real value



Pilots often stall because organisations treat AI as a replacement for engineers rather than an amplifier of experienced teams.

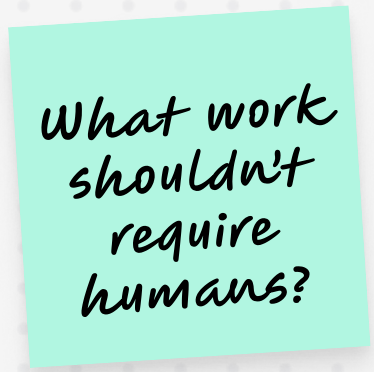
AI delivers the greatest impact when it works alongside senior engineers who understand the architecture, provide AI tools with the right context, and manage the trade-offs involved in building reliable production systems.

AI can accelerate many parts of engineering work, from writing tests to automating environment setup. These capabilities are most effective when guided by engineers who set technical direction and take ownership of system design.

Instead of increasing team size, organisations can form lean teams of experienced engineers supported by AI agents and automation. Engineers focus on complex problems and structural decisions, while AI handles repetitive work across the delivery pipeline.

Research supports this model. GitHub's 2023 Copilot study found developers completed coding tasks up to 55% faster with AI assistance. **McKinsey estimates generative AI could improve developer productivity by 20–45% across certain engineering activities.**

The result is higher throughput without sacrificing technical discipline or system reliability.



03_ Use AI to solve the problems that have stalled for years

Many organisations carry large-scale technical work in existing systems that remains unfinished because it is too complex or time-consuming to prioritise, particularly in brownfield systems.

Examples include:

- Migrating legacy platforms to modern frameworks
- Upgrading major dependencies across large codebases
- Retrofitting test coverage to legacy systems
- Removing years of accumulated technical debt
- Refactoring inconsistent code structures

These problems are usually well understood, but the effort required to solve them manually is difficult to justify alongside feature delivery.

AI makes these problems more tractable. By analysing large codebases, generating tests, identifying patterns, and proposing refactors, AI can accelerate complex development tasks that previously required extensive manual effort.

Teams can therefore address structural problems that would otherwise remain unresolved.

Solving these long-standing technical challenges often produces the most meaningful results. Platforms become easier to maintain, delivery pipelines become more reliable, and technical debt begins to shrink.

Applied to these problems, the impact of AI becomes visible not just to developers but to the organisation as a whole.

AI-native engineering turns AI pilots into delivery impact

Ineffective technology is rarely the reason AI pilots stall. They stall because experimentation never evolves into changing how software delivery systems operate.

Many pilots succeed technically but fail to move into production when security and governance concerns emerge. AI systems interact with sensitive data, internal services, and decision processes. If access controls, auditability, and risk management are not designed in from the start, organisations often postpone deployment until these gaps are addressed.

Conversely, the organisations moving beyond pilot mode apply AI where it directly influences delivery outcomes, particularly within existing systems where constraints are most visible. They **start with the work that slows teams down, combine AI with experienced engineers, and use automation to tackle complex technical challenges that block progress.**

This is the principle behind AI-native engineering. Instead of layering AI onto existing workflows, it embeds AI into software architecture, delivery processes, and governance from the start.

The result is measurable improvements in how software is built, delivered, and operated, with security built into the delivery system rather than a barrier discovered at the point of release.

Where AI is helping and where it is not

If AI is increasing output but engineering performance is unchanged, the issue may not be the tool.

Our **60-minute AI-native engineering** Inspire session shares what we're seeing across enterprise environments — where AI is improving delivery outcomes and where it is introducing friction.

Book a session to get a clearer perspective on how delivery systems are evolving and where AI can drive meaningful impact in your organisation.

Book your session

nearform.com

Nearform