lessons learnt

running

AI-native engineering

in 2026

INTRODUCTION

We're not analysts describing AI from the outside, we're the people who spent the last year actually building with it: on live systems, for real clients, with fixed deadlines.

Our Field Manual in 2025 laid out where to start. A year on, here are 9 things we now know that we didn't then, some hard-won, some still not fully solved.

LESSONS LEARNT

<Small teams_big leverage>

LESSON 01

A few senior people plus AI beats a bigger team, every time we've tried it

The best results we've had all year came from right-sizing teams and leveraging AI, not from adding more headcount.

Why it matters_<impact>

The instinct when scope grows is to add more people. Every case we can point to says the opposite works, small and senior beats big, provided the team is genuinely right-sized for the job and everyone around it in the delivery lifecycle is working AI-native too. Skip that second part and you just move the bottleneck instead of removing it.

Nearform in practice_ <in the field>

One engineer rebuilt a major insurance provider's legacy database instead of a full team. A compliance project for a state-owned service provider took four people, six weeks. A national postal service's customer-facing AI agent project was pre-scoped at seven people and delivered with three.

In their words_

“Large teams introduce coordination overhead. Decision-making slows. Context is lost across layers. What was once a strength, headcount, becomes a drag on delivery.”

Ciaran Cosgrave, CEO, Nearform

<Build fast_decide slow>

LESSON 02

Building got faster. Deciding what to build did not

Code takes days now, not weeks. Deciding what to build, and proving you built it right, still doesn't.

Why it matters_<impact>

Building used to be the slow part, writing the code and getting it working, so it made sense to deliberate before starting. That's flipped, but the two ends of the process haven't caught up. Getting stakeholders aligned on what to build, and getting it validated and signed off afterwards, still moves at the pace of people's calendars and compliance, not engineering.

Nearform in practice_ <in the field>

A workshop that captured 70 requirements for a customer-facing AI agent project led to a working demo in three days instead of four weeks. On a fintech startup's platform, the team were able to rapidly build and iterate on prototypes up to 15 times in the first few weeks, meaning that the client could review working software as opposed to debating features in requirements documents.

In their words_

“The real value now is collecting requirements and aligning stakeholders, because the code itself is cheap. You get something in front of people fast, and that's when you find out if it's actually what they wanted.”

Alfonso Graziano, AI Tech Lead, Nearform

Map the system and understand it before you let AI touch it

Point AI at a system nobody's mapped, and it won't ask questions, it'll guess. Confidently. And it'll be wrong in ways that look right.

Why it matters_<impact>

Legacy systems run on knowledge that was never written down, why a rule exists, what a workaround was for. AI reading that code cold doesn't know what it doesn't know, so it fills the gap with a plausible answer instead of flagging the gap at all. That's a different, more dangerous failure mode than a human making a mistake, it's confident, well-formatted, and it passes its own tests.

Nearform in practice_ <in the field>

On a legacy database rebuild for a major insurance provider, an engineer read the existing system's behaviour and turned it into a specification before writing a single line of new code, then checked that spec against the original before trusting it.
On a separate modernisation project, the same discipline dropped codebase mapping from two weeks to hours, and let the team ship six months of scope in eight weeks, four times faster than another consultancy quoted for the project.

In their words_

"AI agents working on legacy code without a map produce dark code, output that looks clean but quietly breaks things nobody knew were connected."

Katie Roberts, Tech Director and AI-native Specialist

<Team alignment_one approach>

LESSON 04

As a team, align on the same engineering approach

A team split between AI-native and hand-coding-by-choice doesn't get the best of both, it creates friction.

Why it matters_<impact>

This isn't a technology question, it's a team workflow one. Running two philosophies side by side sounds like flexibility. In practice it creates disagreement over what “good” review looks like, and leaves gaps in the documentation wherever engineers are still coding by hand.

Nearform in practice_<in the field>

On a fintech startup's platform, one team ran a trial mixing spec-driven and hand-coding, then made the call to commit fully to spec-driven.

In their words_

“There are no benefits to this mix.
It's better to go AI-native and trust the process.”

Piotr Zimoch, AI Tech Lead, Nearform

Match the rigour to the risk: not every change needs the same process

Going all-in on spec-driven development a year ago sounded like the safe, disciplined choice. It wasn't the whole picture.

Why it matters_<impact>

The instinct was to treat spec-driven development as all-or-nothing, you were either doing it properly or you weren't. The real distinction isn't which framework you're using, it's how you're asking for the change: write a full spec and decompose it into tasks for substantial work, or just prompt directly for something small and well-defined. Confusing the two is what makes teams either over- process trivial work or under-specify risky work.

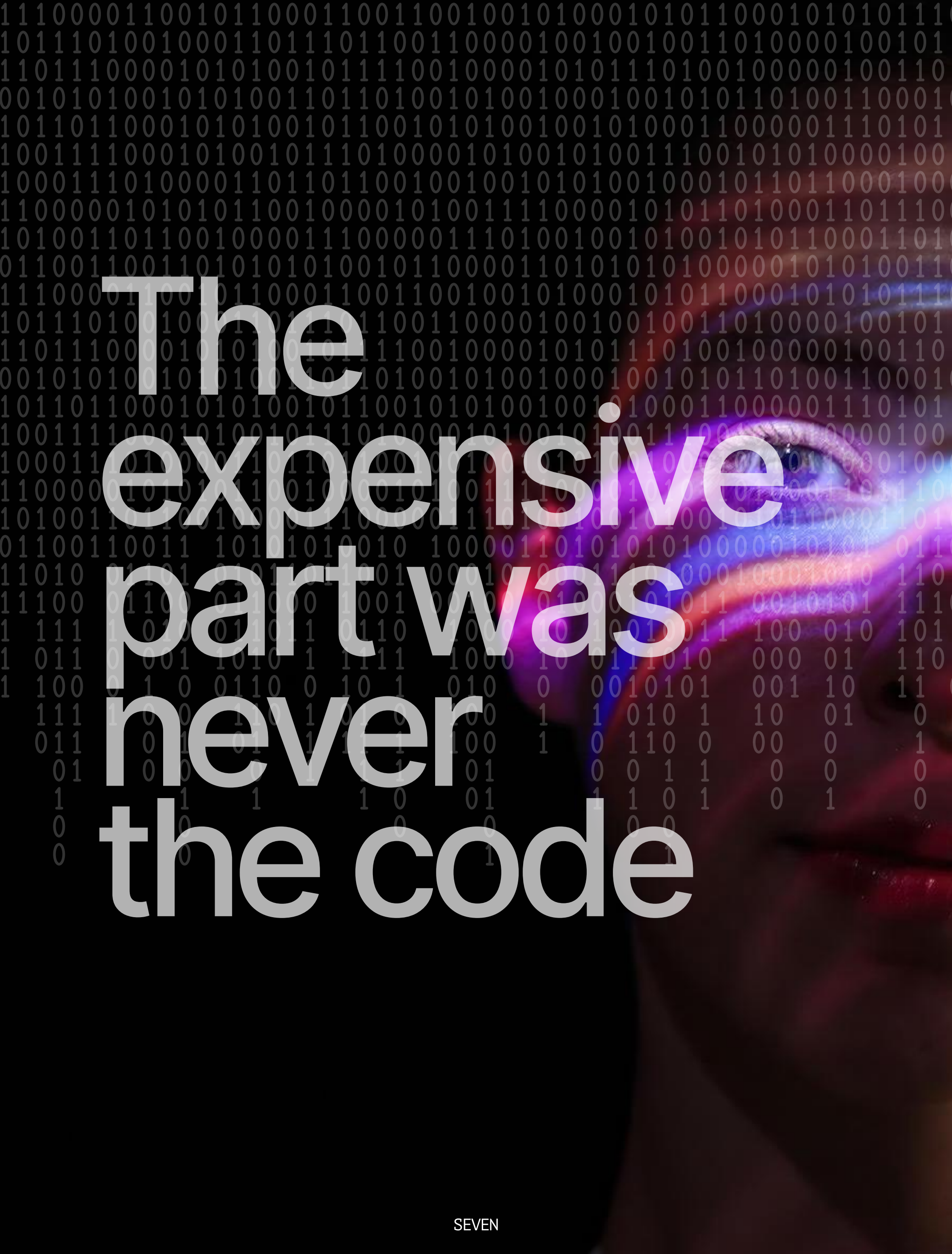
Nearform in practice_<in the field>

On a specced project, only the part of the spec that actually changes gets touched, a small tweak doesn't trigger a full rewrite.

In their words_

“Smaller changes don’t always require long-winded specs. The base harnesses we all interact with have become so capable, it’s now about the right tool for the right job.”

Cian Clarke, Head of AI, Nearform



SEVEN

<Unwritten knowledge_real cost>

LESSON 06

The knowledge nobody wrote down becomes the real cost

Out of the box, AI has no idea how your business actually works, and finding that out mid-project is expensive.

Why it matters_<impact>

An AI project is often the first thing that exposes how much institutional knowledge only ever lived in people's heads. What a field really means, why a process works the way it does, none of it was written down because it never needed to be, until now.

Nearform in practice_<in the field>

On one client's system, with 330,000+ documents across roughly 30 indexes, the obvious technical approach demoed beautifully and collapsed at scale. Field names and types carried no meaning: what was comparable, what needed deduplicating, what a date actually represented. The fix was one human-written description per field, read by the agent at query time, with a CI check that fails the build the moment a description goes missing or drifts out of date.

In their words_

"Codifying institutional knowledge unlocks huge new opportunities – but the challenge becomes scaling that operation, and keeping this information up to date. Here, agents can be our friend and enforcer, ensuring these definitions aren't stale."

Cian Clarke, Head of AI, Nearform

<Token economics_cost control>

LESSON 07

Token economics is rarely solved by swapping in a cheaper model

The fix for a runaway AI bill usually isn't a cheaper model, it’s finding out which part of the system is burning the budget, then right-sizing that part with evals to prove you didn't lose accuracy.

Why it matters_<impact>

The instinct is to assume a big bill means you need a smaller model across the board. Swapping blindly trades away accuracy for a saving that often wasn't where the cost was. And a cheaper model can look like a regression when the real problem is a prompt still sized for the larger one.

Nearform in practice_<in the field>

On a production multi-agent system, P95 cost per trace fell from $2.40 to 50 cents, with overall running cost down over 70% and accuracy holding (or even improving).

The starting point was telemetry, not model selection: one agent was consuming about 65% of the system's tokens, so that's where the work began. Model right-sizing was part of it, done per agent rather than globally, one narrow-job agent dropped ~90% in cost from a single swap. The rest came from trimming tool output before paying for it, optimising caching, and moving mechanical work out of the model and into sandboxed code. None of it was safe to ship without evals against a golden dataset proving the saving hadn't cost quality.

In their words_

“Reducing cost by almost 80% felt like an impossible challenge when we started. We managed to do it using a mix of techniques, and every change was supported by our evals, allowing us to reduce the cost while keeping accuracy high.”

Alfonso Graziano, AI Tech Lead, Nearform

LESSON 08

Trust has two faces: proving it's safe, and trusting it to run itself

This year we had to earn trust twice, once with regulators and clients, once with our own engineers.

Why it matters_ <impact>

These are different problems that both happen to be called “trust”, and they need different evidence. Proving a system safe to an outside party takes an auditable trail from request to decision, no shortcuts. Earning your own team's confidence to let it run unsupervised is built on track record, not on assertion.

Nearform in practice_ <in the field>

On a state-owned service provider's compliance project, the hard part was never proving the AI could analyse data quickly, it was proving every recommendation was explainable, auditable, and signed off by a person, the same standard you'd hold a human analyst to. Separately, inside our own delivery teams, a year ago every agent action needed manual approval. This year, better models, sandboxed environments that limit what an agent can actually touch, and a year of experience using both mean engineers now let agents run unsupervised for stretches at a time. The permissions changed because the containment did, not because we asked more nicely in the prompt.

In their words_

“Last year you'd have been a fool to give an agent full permissions. What changed is the models got better and safer, and the harness around them got better too.”

Alfonso Graziano, AI Tech Lead, Nearform

The tooling was never the constraint

Shipping isn't the same as people using it

A tool that works and a tool that gets used are two different achievements; measure both.

Why it matters_<impact>

It's tempting to treat “we built it” as the finish line. The harder, more honest measure is whether the people it was built for actually adopted it, and that's not guaranteed just because the technology works.

Nearform in practice_<in the field>

For an executive search consultancy whose AI report tool cut generation time from 45-plus minutes to a fraction of that, adoption still came in at 60%, tracked separately from the speed gain. The same pattern showed up for a global life sciences company: the real struggle was never the technology, it was getting people to use it. What closes that gap is on-the-job practice, not courses, the habit only sticks when people use the tool inside real work, with someone senior enough to correct them when it goes wrong.

In their words_

“We can ship something that works in a week now. Getting people to actually change how they work around it still takes months. The tooling was never the constraint, the habit is.”

Cian Clarke, Head of AI, Nearform

If you're running AI-native projects of your own and hitting the same walls, we'd like to hear about it, and we're happy to talk through what's worked for us, what hasn't, and why.

Get in touch at **hello@nearform.com**