One pull request.
Six lines of code.

24

days open

0

review comments

2

files, one a deletion

Opened 6 March. Merged 30 March. Nobody disagreed with it, nobody asked a question, nobody objected. It waited.

Pending Approvals: An AI Story

Pedro Vicente

How we started

OUTCOME-BASED

We committed to a result, not to a number of people for a number of months. Every month over ran on us.

THE PRODUCTIVITY BET

Mindera assumed AI would optimise delivery — and that assumption was priced into the speed/time we signed up to.

THE SCOPE

Applied AI on Microsoft AI Foundry — building an AI feature. It didn't implied we would use AI to build it which implied changing ways of working was out of scope.

The (initial) squad, in facts.

5

engineers, one QA, one EM

3

shared repositories, each owned by another team

1

repository we owned — the only one with no approval step

Applied AI

feature - customer-facing, built on foundation models

Alongside

set up beside the existing teams, to add capacity without disrupting them

What "AI-first" actually meant.

Before any of the numbers mean anything, here is the practice they came out of.

01

The tooling

Copilot

02

Applied AI models

Microsoft AI Foundry

03

The daily loop

Jira
Figma
Copilot prompting

What "AI-first" actually meant.

Before any of the numbers mean anything, here is the practice they came out of.

01

The tooling

Copilot

02

Applied AI models

Microsoft AI Foundry

03

The daily loop

Jira
Figma
Copilot planning

What "AI-first" actually meant.

Before any of the numbers mean anything, here is the practice they came out of.

01

The tooling

Copilot

02

Applied AI models

Microsoft AI Foundry

03

The daily loop

Jira (Atlassian MCP)
Figma
Copilot planning

What "AI-first" actually meant.

Before any of the numbers mean anything, here is the practice they came out of.

01

The tooling

Copilot

02

Applied AI models

Microsoft AI Foundry

03

The daily loop

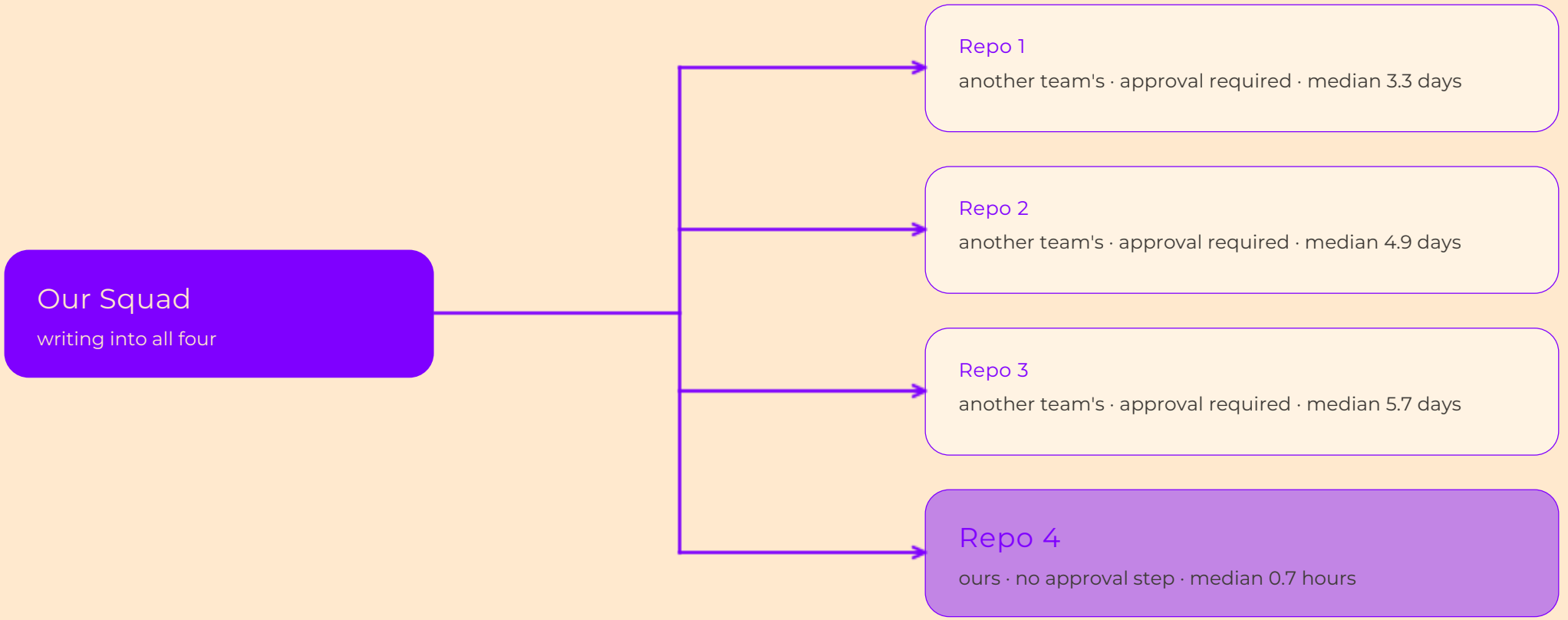
Jira (Atlassian MCP)
Figma (MCP)
Copilot planning & Implementation

KICKOFF DECK, SLIDE ONE

“We will not interfere with existing ways of working.”

I wrote that sentence myself. It is the one that killed us.

Three repositories we didn't own. One we did.



Medians. 172× apart — and the fair objection to that comparison gets its own slide later.

What the bet cost.

5→7
engineers

5→8
months timeline

+27%
cost overrun

Three months of calendar cost twenty-seven percent of the budget. Not one of those months was spent writing code.

The schedule ran 60% over. The cost ran 27%. On a fixed-price outcome, that gap does not disappear — it moves onto our side of the contract.

Outcome

What made us proud... and what didn't

−80%

approval latency, 10.4 days to 2.2

17×

monthly PR output, 3 to 52

2.2

days to approval, against 4.1 from other teams

Planned 5 months. Took 8.

By the end we were the fastest team in that codebase. It still ran three months over.

Production got ten times faster. The organisation around it did not move at all.

WHAT WE MADE FASTER

Writing a feature — 2 hours

Merging where we owned the repo — 0.7 hours

Monthly output — 3 to 52 pull requests

WHAT DIDN'T MOVE

Cross-Team reviews — 2.9 days

Getting into the systems — weeks

Legal and platform sign-off — the final weeks

AI doesn't remove your bottleneck. It relocates it — into whichever part of the organisation nobody has ever measured.

Five reasons the bet was naive.

01

Access

Getting the team into the systems took weeks, not days.

02

Ramp-up

Senior engineers are not AI-first engineers on day one. The curve is real.

03

The models

A year ago the models were nowhere near as robust as they are today.

04

The queue

Tickets sped up. Then they stopped, in front of other people's review.

05

The approvals

Legal and AI Foundry governance, in the final weeks. We were the first.

Three mistakes that were mine

MISTAKE ONE

AI is a skill, not a switch.

WHAT I ASSUMED

That senior engineers would become AI-first developers as soon as we trained them.

AI IS A SKILL LIKE ANY OTHER TOOL

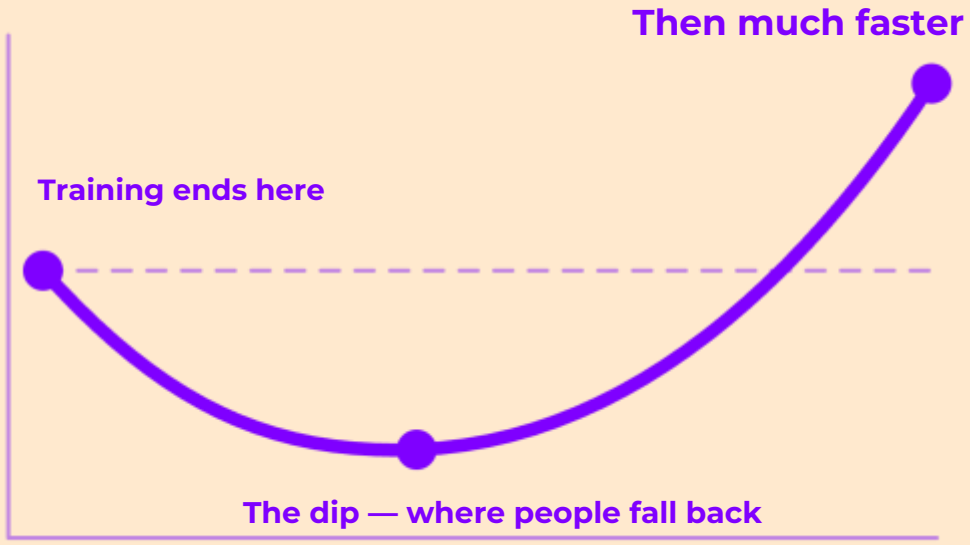
It takes knowledge and practice. Kicking off the project and upskilling as we went was a far bigger risk than I priced in.

UNDER PRESSURE, PEOPLE REVERT

As the pressure mounted, people fell back on what they were comfortable with — it feels faster than risking something they aren't.

IT NEEDED CONTINUOUS COACHING

Not a workshop. And the curve makes a team slower before it makes them much faster — the dip lands exactly when the deadline does.



The dashed line is the pace they already had. Nobody budgets for the part below it.

MISTAKE TWO

I promised not to change how we worked.

SLIDE ONE OF THE PROJECT

“This will not interfere with other team's existing ways of working.”

POST MORTEM

“The ways of working broke all our assumptions.”

MISTAKE THREE

I did the sum on the wrong number.

Writing the code	2 hours
Collecting two approvals	two weeks

WHERE THE TWO WEEKS ACTUALLY WENT



Medians across the three shared repositories. Approval-to-merge ranged 0.8–10 hours.

Five pending approvals

Five things that waited. Each one turned out to be measuring something.

#1 — Day 1 access to the model.

60

days before we could begin

I ASSUMED

Everything was in place

WHAT WAS ACTUALLY HAPPENING

No precedent for the work. No clear governance. A consumption cost model that fit no budget line. Four functions, none of whom had done this before, each waiting on the others.

WHEN IT HIT US

Before we could write a line. Day one to day sixty, with the team already hired.

#2 — The pull request.

24

days for six lines of code

Zero review comments in all of them

I ASSUMED

The reviewers were busy, or slow, or territorial.

WHAT WAS ACTUALLY HAPPENING

They maintained a system that had technical debt, and our change touched a part of it where doing the job properly meant a refactor neither team had the budget or the time for.

WHEN IT HIT US

The middle of the project — exactly as our output started to climb. The faster we shipped, the longer the queue.

#3 — Legal has entered the chat

0

frameworks to sign off against

Same for penetration testing. Same for performance standards.

I ASSUMED

Legal being risk-averse.

WHAT WAS ACTUALLY HAPPENING

No precedent, no internal policy, no regulator guidance they trusted, and personal professional exposure if they got it wrong.
We were helping draft AI policy, and we had budgeted for none of it.

WHEN IT HIT US

In the final weeks, before go-live. Nobody in the company had been asked these questions before.

Every hour Legal spent on us is an asset that company now owns. We expensed it to the wrong project.

#4 — AI Foundry Configuration

8 : 800

people affected, against the forum's remit

A change to our own tooling, decided org-wide.

I ASSUMED

It's our tooling. We change it. Small, reversible, ours.

WHAT WAS ACTUALLY HAPPENING

Because it was new, policies were being set in place for the entire organization. A change affecting eight people needed agreement from a forum designed for decisions affecting much more.

WHEN IT HIT US

In the final weeks, with the feature built. We were first, so we paid the price of being first.

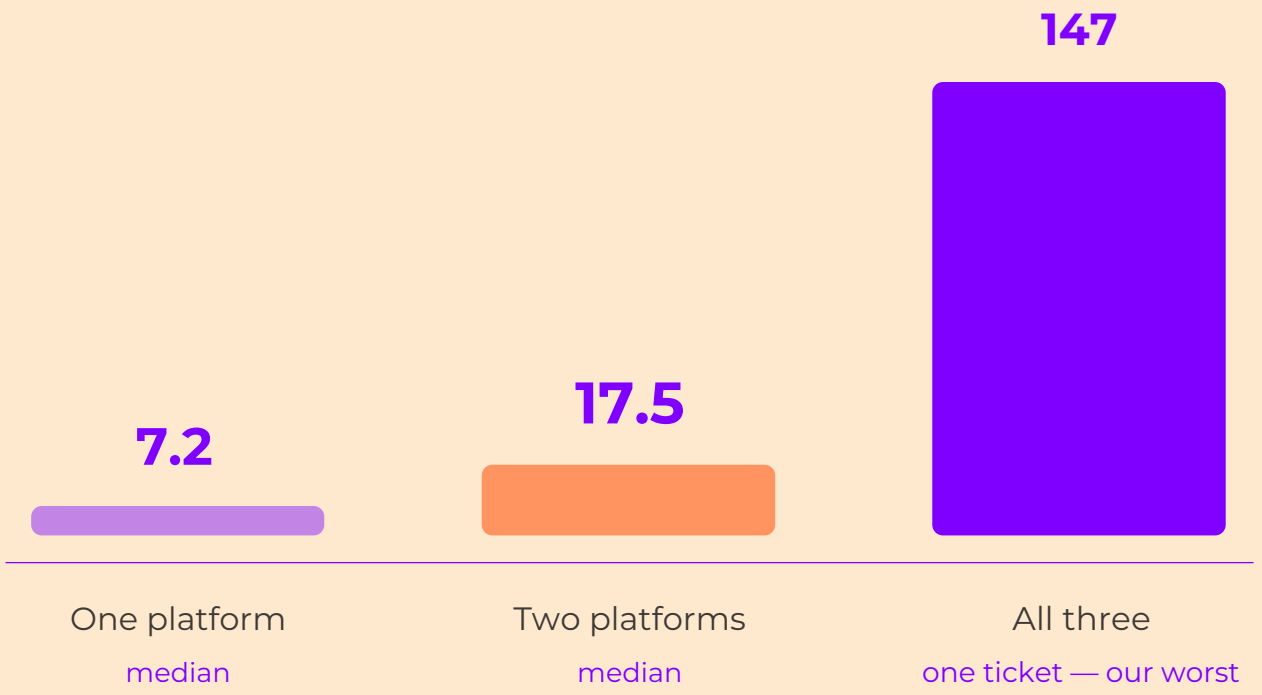
IF YOU TAKE ONE THING

Every genuinely new capability starts with an undefined blast radius. And undefined defaults to everyone.

This has nothing to do with AI.

#5 — Dependencies hit us hard

Establishing which internal API we needed took one prompt. Then who owned it and could agree on a path, took **15 days**.



Days to close a ticket, by how many of the three mobile platforms it touched. The first two are medians; the third is a single ticket — the worst one we had.

None of the five were AI problems.

The problems were there already. We didn't prepare for AI.
AI just amplified them.

Learning the hard way

I was wrong about almost everyone on the other side of the queue.

THE SENIOR ENGINEERS

I ran workshops on AI.
What they needed was an answer to what my role now.

Everyone was right. The pace wasn't.

Legal

No precedent, no policy, no regulator guidance, personal exposure. Making sure it was done properly is the job.

Their cadence was built for decisions that hold for years. Ours changed weekly.

The owning teams

They were protecting a codebase they would still own in two years, and doing a review they could stand behind.

Their review pace was set when a team touched that code once a quarter. We arrived weekly.

Nobody was wrong about the work. The mismatch was speed — and we were the ones who changed it.

Nobody in this story was incompetent or obstructive. The system worked exactly as designed.

Was the code any good?

0

bugs found in the two months after go-live

14→2.3

review comments per PR, first full month to last — converging on the organisation's own baseline while output rose 17×

71→93%

clean approval rate — approved with no changes requested

Clean go-live — still all green three months later.

What moved those numbers.

The quality did improve across the eight months. None of it came from the tooling getting better.

01

Time in the codebase

The engineers learned the systems they were writing into.

02

Inter-squad collaboration

To use AI properly we need to adapt ways of working

03

Continuous coaching

AI-first is a practice to be kept up, not a workshop to be attended, and ran the squad that way.

04

AI Champion

We replaced one engineer with a more AI-savvy one, whose job was to push AI-first inside the squad.

Half our output went through review that took days.

4.6

review comments per pull request in the three shared repositories, and days of waiting on each one.

34% of Android and 42% of the shared KMP pull requests sat for more than a week.

The other half went through almost no iteration at all.

99.5%

of pull requests in our own repository had close to zero review comments.

57% of ours merged inside an hour.

The fair reading of 172×.

THE OBJECTION

0.7 hours against 5.7 days isn't fast code versus slow process. It's unreviewed code versus reviewed code.

AND IT'S FAIR

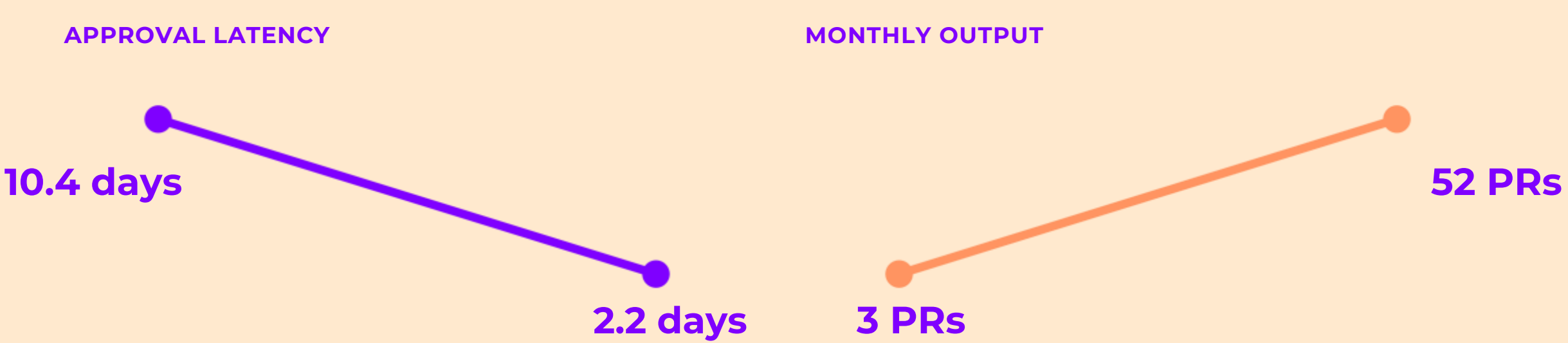
7 of 183 pull requests in our repository ever received an approval. 99.5% had no review comments. They ran 240 lines against 90.

WHAT STILL HOLDS

Same engineers, same tooling, same eight months. The only variable is whether a change had to cross a team boundary. That's the number I'm asking you to look at — not the depth of the review.

What changed

This is the chart we were proud of.



In the final month we were approved in 2.2 days against 4.1 for everyone else in the codebase — faster than any other team writing into it.

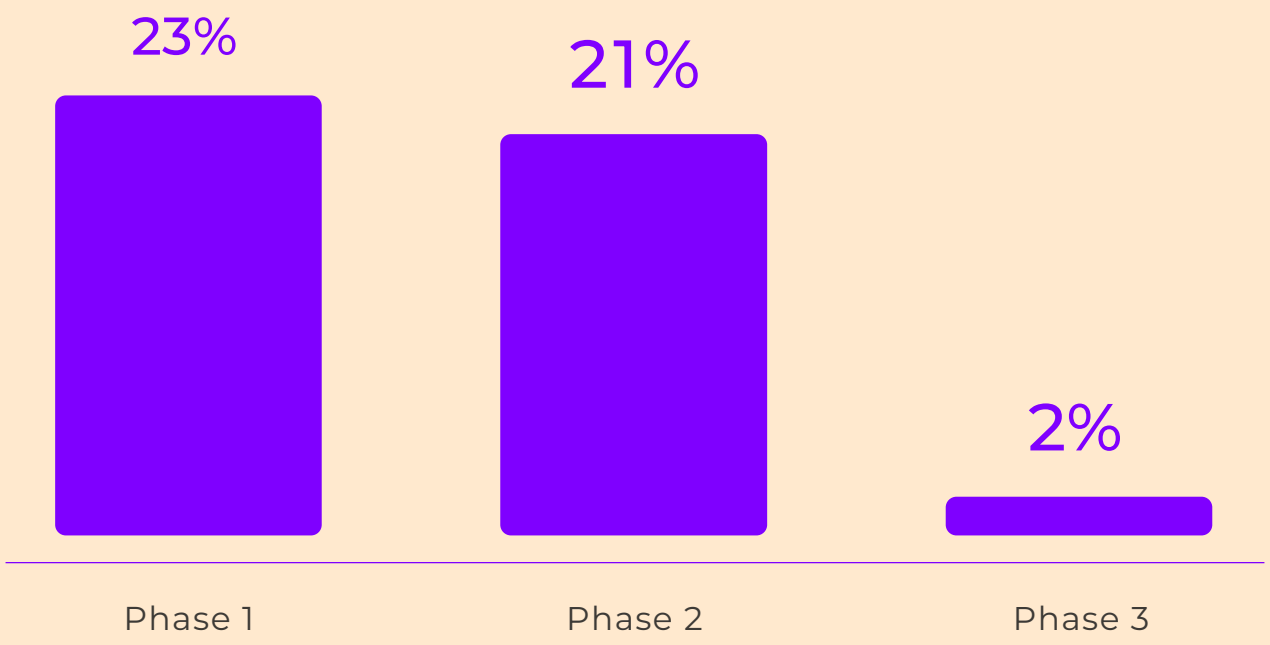
The interesting part isn't the line. It's where it bends.

First month to last, eight months. Endpoints only — the phase detail is on the next slide. Over the same period internal approvers went from 3% of approvals to 57%: part of this fall is who was signing, not how fast they signed.

Three phases.

Phase	Approval latency	What moved
1 · One engineer per platform	8.5 days	Ticket cycle 18 days end to end
2 · Team increase by two and replaced one by an AI Champion	Unchanged — slightly up	Throughput tripled, ticket cycle halved
3 · Ways of working alignment to improve ways of working	Down 58%	PRs waiting over 14 days fall to 2%

The tail collapsed.



Share of pull requests waiting more than fourteen days.

What worked took "one day"

5 months

Escalating. Process changes, governance forums, meetings about meetings.

"1 day"

Sitting down with all engineers and agreeing on improved ways of working

This was really a build-up of trust, until the engineers agreed to change their own ways of working and automate more.

Caveats, out loud: roughly 30 pull requests per phase · phase boundaries are approximate · internal approvers went from 3% of approvals to 57%. Part of the speed-up is who was approving, not only how fast.

Governance is a way of working too

So we did it — approval latency down eighty percent, fastest team in the codebase. Then the thing we had built on disappeared.

And then the model was retired.

MODEL RETIREMENT NOTICE

The model we had built on was retired. The feature broke.

We managed out before go-live rather than after. We got lucky — it was a matter of weeks.
We manage to request new models and reploy within two days, with the majority of time being the request.

What we had, and where it broke anyway.

HAD	**Abstraction layer**	We could swap to any other model available on AI Foundry without touching the feature code.
HAD	**Evaluation suite**	Behaviour changes were caught by tests, and we checked for prompt injection.
GAP	**Direct model calls**	Foundry didn't carry everything we needed, so some models were called directly (fallback were not available)
GAP	**Lifecycle governance**	Ownership existed — a council convened whenever a need came up. What didn't exist was a process guaranteeing that a model change or removal reached everyone affected, early enough to prepare.

One of those direct models was deprecated without notice. Boom.

Put two numbers next to each other.

Getting permission to use a model



60 days

Our median ticket, end to end



under 5 days

Permission cost about twelve times more than our median feature — and it never appeared on a board, nobody was assigned to it, and it has no cycle time.

And that model was retired before we shipped.

Our approval process was slower than the lifecycle of the thing it was approving.

Ways of working don't stop inside the team. Governance is part of the drill - and in this case it's built for decisions that outlive the thing that they govern.

What expires, and what doesn't.

HAS A SHELF LIFE

The specific model limitations we hit

How much hand-holding the tooling needed

That these skills were genuinely scarce at the time

Don't learn from any of it.

DOESN'T EXPIRE

Decision rights are undefined for anything unprecedented

Speed differentials between teams create queues, not throughput

Status threats to senior practitioners are not solved by training

Governance vacuums get paid for by whoever arrives first

Latent dysfunction stays invisible until something moves faster than it

The models are better now. None of the second column has improved at all.

Here's what I can't prove.

01

How much faster without the gates?

No comparable team ran the same work through fewer approvals. I can't tell you what we would have delivered without it.

02

How much faster than a team without AI?

No baseline team. Everything I have compares us against ourselves over time.

03

What the 17× is actually measuring

Pull requests, not value — and three a month is a ramp-up baseline, not a pre-AI one. Part of the 17× is starting from nothing.

And our pull requests were bigger: 240 lines median, against 90 in the shared repositories.

Some of that seventeen-fold is batch size, not speed. I am not going to pretend otherwise.

Three lessons.

ONE

Coaching is not optional

A workshop starts the clock. Practice is what moves it. Budget continuous coaching, or budget the dip — you are paying for one of them either way.

TWO

AI only pays if the way you work changes with it

We bought the tool and kept the process. The process set the pace, and the tool had nowhere to put the speed it gave us.

THREE

Alignment is not only yours to do

Reviewers, Legal, the platform, governance — everyone you depend on has a their own pace. **You can only move as fast as the slower.**

We changed how the team worked. **We never changed the ecosystem the team lives in.**

Five things you could do on Monday.

01	**Fund coaching, not a workshop**	Name who coaches and for how long. Budget the dip before the curve turns up.
02	**Change the way of working first**	Decide what AI changes about review, testing and estimating — and write it down before you start.
03	**Map who you depend on, and their pace**	Every approver, reviewer, platform and forum. Their cadence is the ceiling on yours.
04	**Collect the end approvals at the start**	Access, Legal, the platform. If you are first you are writing the policy — fund it separately.
05	**Treat governance as a way of working**	If a decision outlives the thing it governs, the process is the defect, not the exception.

MINDERA AI INSIGHTS

WHERE WE STARTED

Six lines. Twenty-four days.

The most painful measurement I've ever written about a project.

Have you thought about whether you have this kind of gap?
Have you measured it yet?

Questions.

Pedro Vicente