Friction

The Silent Killer

Christoffer Råsten

TROi

Friction — The Silent Killer
More Value From What You Already Have

Written by Christoffer Råsten

christoffer.rasten@troi.se
Print Edition 1.0
Based on the online edition as of 13 August 2026

Organizational Flow is a living paper that continues to evolve online. This print edition represents a snapshot of the text at the date shown above.

The current edition is published at
https://troi.se/organizational-flow/

TROi — independent architecture and organizational design

Publisher: BoD · Books on Demand, Östermalmstorg 1, 114 42 Stockholm, Sweden, bod@bod.se
Printed by: Libri Plureos GmbH, Friedensallee 273, 22763 Hamburg, Germany

Contents

PART II — SEEING ORGANIZATIONAL FLOW

Preface

For years I introduced myself as an Enterprise Architect because that was my role, and for the most part it described the work I was there to do.

My focus was never to study organizations for their own sake. It was to help deliver change, solve problems, design systems, and make technology work a little better than it did yesterday.

Along the way, though, I kept noticing something that seemed to matter just as much as the technology itself. Similar decisions became easy in one place and surprisingly difficult in another. Teams with comparable skills achieved very different outcomes. Sometimes everything seemed to flow almost effortlessly; other times even simple things became unexpectedly hard. The technology explained some of it, but rarely all of it.

At first I treated those observations as part of the scenery. Every organization has its own history, its own people, its own ambitions and constraints. That is exactly as it should be. But after enough years, enough companies and enough conversations, I started recognizing familiar patterns. Different industries. Different terminology. Different operating models. Different explanations. Yet many of the underlying mechanisms seemed remarkably similar.

Some of it went into a blog along the way, which is still there with its dates on.

I wasn't looking for a theory, and I'm still not sure that's what this is. If anything, this paper is an attempt to make sense of something I kept encountering in practice. Architects describe part of it. Leadership researchers describe another. Systems thinkers, psychologists, organizational designers and many others all contribute valuable perspectives. I

learned from all of them, but I was left with the feeling that they were often describing different parts of the same system.

Over time, I started to think of what I was observing as Organizational Flow.

It is the phenomenon rather than anything I built — and it turned into a lens for understanding why some organizations can turn knowledge, decisions and effort into value with far less friction than others. Not because I believe it explains everything, but because it gave me a way to connect observations that had previously felt unrelated. It is still a working theory. I expect parts of it to change. I hope they do. Every conversation, every challenge and every reader who points out something I have overlooked helps sharpen it.

A word about the title, since it is the first thing you saw. *The Silent Killer* is a bit much. It sounds like a documentary about a gas leak, and I did wonder whether a paper on organizational design ought to sound quite so much like an airport paperback. I kept it because it happens to be accurate — nothing dramatic occurs, nobody raises an alarm, and one day you look up and a year has gone past. But I take the point, and if you read the whole thing and conclude the friction is merely quite unhelpful rather than homicidal, I will not argue.

If there is one invitation in these pages, it is simply this: start paying attention to where work flows easily, where it slows down, and what seems to make the difference. You may not draw exactly the same conclusions that I have, but I hope you begin to notice some of the same patterns.

This paper is my attempt to describe them.

EDITOR'S NOTE

This has taken a while to get down, and I could keep tinkering with it privately for another year without it getting much better. Putting it somewhere people can actually read it seemed more useful. What it needs from here is the kind of input I can't give it on my own.

A Note to the Reader

What follows is my own reading of what I have seen, shaped by years of working with organizations.

Many of the examples and stories come from my own experience. They are how those situations looked to me at the time, and what I took from them. Other people who were there may well remember them differently, which is only to be expected.

The thinking behind all of it has been shaped by conversations with colleagues, clients, mentors and friends, and by more books, courses and conferences than I could sensibly list. Anyone who has spent years in a profession knows how hard it is to say where one idea ends and someone else's begins.

So if something here feels familiar, that is probably why. I am grateful to everyone who has contributed to it, knowingly or otherwise. Where I have got something wrong, that part is mine.

The Ten-Minute Version

Everything worth knowing, before you decide how much more you want.

Foresters have a word for a ring cut through the bark, all the way round the trunk, that stops the sap without felling the tree. **Girdling.** The canopy stays green for weeks. From the road it looks like a perfectly good tree.

I have walked into organizations in exactly that condition. Profitable, busy, shipping — and nothing anyone decided was reaching anything. Nobody could see it, because everything visible looked fine.

So the question this paper keeps returning to is not what is broken. It is: how much more could you do with what you already have? With the people in the building today, before the next hire, the next platform, the next transformation.

In most places I have worked, the honest answer is quite a lot more.

Here is the smallest version of why.

Somebody sent a message on Tuesday morning. A short one — a question they needed answered before they could carry on.

The answer came Thursday afternoon. Nobody did anything wrong. The person answering was in workshops, it wasn't marked urgent, and two days is a perfectly civil response time.

Two days is also two days. It shows up in no system, on no report, in nobody's numbers — and if you asked either of them about it a week later, neither would remember it happened. Now multiply that by everyone currently waiting on somebody else, which is very nearly everyone.

That is the thing I kept noticing, across twenty-five years and a lot of very different organizations. Not incompetence. Not bad people. Just a slow, invisible tax nobody ever agreed to pay.

I have been writing down what I found. It is now past fifty chapters, which was not the plan. The plan was “a few observations.” Somewhere around chapter nine it became clear the plan and I had parted ways amicably.

So: observations, mostly, plus a working theory of what to do about them if you would like digitalization to deliver what everyone keeps promising it will. And a certain amount of irony in having written that many chapters about friction, which I am aware of.

Which is why this page exists — along with the audio, a set of questions that tells you which chapters are actually yours, and chapters built to be read in any order at all. I try to live as I learn, applied to my own material. It seemed only fair to remove some of your friction before asking you to read about it.

Here is the short version.

The idea

Every organization already has a property I call Organizational Flow: how easily information, decisions, execution and learning actually move through it. It rises when that movement is easy and ownership is clear. It falls when friction piles up faster than value gets created.

You don’t install it. You never finish it. It is more like a temperature a building holds — drifting in both directions, all the time, whether anyone is watching or not.

Where it breaks

Almost never in the steps. Nearly always in the seams between them.

Draw a team's boundary around the *work* instead of around the *judgement*, and you get people who can build almost anything and decide almost nothing — so everything escalates, and an escalation is just a handover in a nicer outfit. Split one capability across two departments and a committee appears that nobody remembers agreeing to, because governance is what grows automatically wherever ownership went fuzzy.

"Let me check and get back to you," said politely, four hours at a time, across a handful of boundaries — that alone quietly consumes a full-time role every year that nobody ever hired. Waiting is the sneakier one, because the people waiting never look idle. They have moved on to something else. Which is exactly why nobody notices six months disappearing behind eight small, entirely reasonable three-week approvals.

The trap almost everyone falls into

Getting faster and getting more useful are different skills, and confusing them is expensive. You can speed an organization up beautifully — cycle times down, dashboards green, everybody pleased — while getting extremely good at producing things nobody actually wanted. Fast and wrong is still wrong. Just quicker about it.

What actually helps

Rarely the first thing you would reach for. Sort out who owns what before adding another process, because a new forum papers over unclear ownership without ever settling it. Draw

team lines around a capability rather than a tech stack or a reporting line, and make sure somebody *inside* the team can genuinely decide — not just relay what the real decider probably thinks. And when something is late, ask where it sat waiting rather than why it took so long. One question gets you an answer. The other gets you a defence.

There is no finish line here. It is closer to tending a garden than executing a blueprint — you are never done, you are just keeping the weeds from outgrowing the clearing.

Where to start

Please don't read it front to back in one sitting. I built it specifically so you wouldn't have to, and I would take it personally if you ignored that and then blamed me for the headache.

Each chapter looks at the same idea through a different lens, so the best way in is whichever lens matches whatever is currently irritating you at work. There is almost certainly something. There always is.

Not sure which? Take A Closer Look — eighteen questions about things you can actually observe where you work: who gets to decide, what happens when work crosses a boundary, how long things sit waiting, and whether anything ever gets learned. It hands you the three or four chapters that are actually yours and quietly excuses you from the rest. Nobody should have to reverse-engineer a contents page to find out whether a paper is about their problem.

If you keep one sentence

Technology is rarely the bottleneck. Friction is — and it was never in the steps. It was always in the seams between them.

Fair warning: once you have words for this, it is very hard to unsee. You will start spotting it in meetings, in budget rounds, in org charts and in the gaps between them, and you will become mildly insufferable about it for a few weeks.

But you can start on Monday, and that is the part I would actually press. Pick one capability that matters and ask three people in different parts of the organization who owns it. Don't correct the answers — collect them. It takes an afternoon, it needs nobody's permission, and it produces a fact rather than an opinion, which in a place where the friction is invisible is the only thing that reliably changes anyone's mind.

You will know something by Friday that you did not know today. Everything else here is just more ways of looking.

A Personal Journey

How I moved from writing code to paying attention to the conditions around it.

This is where I come from: technology, and the architecture trade. The part I have enjoyed most, though, has always been where business, people and technology meet. To make sense of the rest of this, it probably helps to know how I got there.

I wrote my first production code in 2001, for an insurance company most people have never heard of unless they've filed a claim through it. AFA Försäkring wasn't glamorous and didn't need to be. It needed systems that worked, and for seven years that was the whole of my job.

By 2008 I'd moved from writing code to designing it — Solution Architect, a title that meant I was now responsible for how the pieces fit together rather than for building one of them. The questions got bigger. Whether a particular function worked mattered rather less than whether the system made sense next to the other twelve it had to talk to.

Five years later the title changed again — Domain Architect — and so did the questions. Less about systems now, more about domains: what belongs together, what doesn't, where one team's responsibility should end and another's begin. I was drawing boundaries I couldn't see, and I called it architecture, because that was the word on my business card.

By 2019 I was Chief Architect and Head of Architecture, leading the function, sitting on the management team, and still working hands-on as an architect. My perspective had shifted. I was no longer responsible only for making good decisions myself, but for creating the conditions in which good decisions could be made throughout the organization.

The throughline across those four titles is simple enough in hindsight. I kept moving further from the code and closer to the conditions the code lived in. What an organization need-

ed to be able to do turned out to matter more than which tools it happened to use, and clear ownership did more good than another layer of governance ever did. None of it was written down anywhere. It was just what seemed to work, learned slowly and one domain at a time.

I had the vocabulary by then. Solution, domain, architecture. What I didn't have was a name for what all four of those jobs had actually been chasing. That came later, and not from anyone in the architecture trade.

A Digital Awakening

An honest question I couldn't answer, and where it sent me.

“Is our IT future-proof?”

Our CEO asked it in a company meeting, the way you ask something you already suspect the answer to. I didn’t have a good answer, and worse, I wasn’t sure what a good answer would even sound like.

So I went looking for people who might. This was around 2015 and 2016 — ten years ago now, which is worth saying, because a good deal of what sounded radical in those conversations is ordinary today and the part that mattered still isn’t. Through a fortunate chain of introductions I ended up talking to Adrian Cockcroft, then at Netflix and AWS, and Rodrigue Schaefer, Head of Engineering at Zalando. Both were generous with their time, and neither said anything I could have found in a book, though I had certainly looked. They arrived at much the same thing in different words: IT should not sit as a separate function. It should be the engine of innovation.

That rearranged something for me. I had spent two decades getting steadily better at running the engine, and almost no time asking why it had been built as a separate room from the rest of the business to begin with.

Once I started looking, the pattern turned up more or less everywhere, including in my own organization. Digital transformation as a rebrand rather than a rebuild — new tools, new vocabulary, the same operating model underneath. The organizations that genuinely changed were the ones where business and technology reinforced each other rather than negotiating a truce.

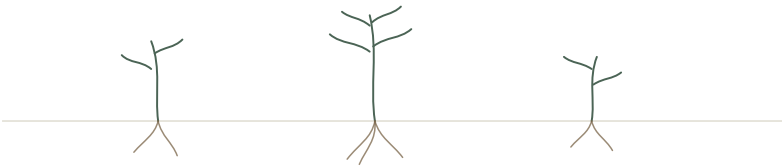
Years later, working across media, telecom and insurance, the tell has stayed much the same. The companies still struggling tend to be the ones treating architecture as a technical

concern rather than an organizational one.

I still think about that meeting, though not because the question was clever. It was honest, which is rarer. Most transformation efforts start with the answer already decided; this one started with someone willing to say they didn't have it.

That is where the argument in this paper came from. It wasn't reasoned out from a theory so much as noticed slowly, in particular rooms, over a long time — worth knowing before reading the rest, because there is a difference between a model someone constructed and one someone kept running into.

Part I is where it stops being a story and starts being an argument.



PART I

Discovering Organizational Flow

How much more could an organization create with exactly what it already has?

Usually quite a lot, and almost never through more effort or more talent. What sits in the way is quieter than that, and it lives in the spaces between people, decisions and teams.

This part is the why, and the frame. What that looks like from the inside, roughly what it is worth, the name I ended up giving the thing worth protecting — and the picture the rest of the paper keeps coming back to.

Introduction

How much more could you do with what you already have?

Every organization is full of things that were once a good idea. The reporting line, the approval step, the weekly forum, the strategy deck nobody has opened since the offsite. Somebody sensible introduced each of them to solve something real, and all of it was meant, in the end, to help the place create more value.

What rarely gets asked is whether it still does. Things brought in for good reasons tend to outlast the reasons, and noticing that is nobody's particular job.

Which leaves a question worth asking more often than it gets asked:

How much more could this organization create with exactly what it already has?

With the people and the knowledge already in the building today, rather than after the next hire or the next transformation.

In most places I have worked the honest answer is quite a lot more, and it is almost never a matter of effort or talent. Everyone is busy, the calendar is full, and yet the thing everybody agrees is important has not really moved since spring — and nobody can quite point to where it stopped.

So a good deal of what an organization is capable of never turns into anything. Structural friction is the name I ended up giving to the mechanism behind that, and the claim the rest of this paper rests on is short enough to say plainly:

Organizations don't become slow because people work slowly. They become slow because structural friction accumulates faster than value is created.

The mechanisms themselves are rarely dramatic. Someone isn't sure whose decision this is, so they check with somebody first. Checking becomes a standing meeting, and the meeting becomes how things are done here. Every step was reasonable, and nobody would defend the result.

One thing worth saying before any of the rest of it.

Friction that somebody chose is not what this paper is about.

Plenty of organizations carry a great deal of it entirely on purpose, and are right to. A bank that puts three pairs of eyes on a payment above a certain size has decided the delay is worth what it prevents. A hospital that will not let one person authorize a change to a dosing rule has made the same trade. Regulated industries are full of friction that somebody sat down, priced, and accepted — and it would be a poor kind of advice that told them to remove it because a paper about flow said so.

There is a third kind, which somebody pointed out to me and which I had been treating as part of the second. Some friction is correct right now because the thing that would remove it does not exist yet. The manual check that is there because nothing checks it automatically. The review that happens because the team could not yet be handed the judgement. That is not carelessness and it is not caution — it is a stage, and stopping the check before building the thing that replaces it would simply be reckless.

The trouble is that this is also the most comfortable explanation available for friction that has stopped being temporary, and it never expires on its own. *We are not ready for that yet* is true for a while and then quietly becomes the reason nobody looks. The version worth holding is narrower: which rung are we on, what would take us up one, and when did anybody last ask.

What I am interested in is the same arrangement arrived at by nobody. The approval that exists because of an incident in 2018 that nobody now remembers. The forum that was set up to solve something since solved. Structurally these look identical from the inside; the only difference is whether anyone can tell you what it is for and what it costs.

So the useful question is never *is there friction here*. There will be, and some of it is load-bearing. It is whether the friction you have is the friction you would choose, now, knowing what it takes out of you — and whether anybody has looked recently.

Organizations are also not interchangeable. What is obviously right in one is obviously wrong in another, which is why nothing in these pages is a recommendation about your particular situation. It is a way of looking at it, and you know things about your organization that I do not.

What I find more interesting is why it went unclear to begin with, because almost nobody chooses it. A reorganization moves a line and a capability ends up split between two teams who each own half of it. Someone leaves and takes the only reliable answer to a weekly question with them. None of it looks like a decision at the time, which is why it so rarely gets revisited.

Organizational Flow

The name I use for an organization's ability to keep creating value is **Organizational Flow**.

It is a property every organization already has, sitting somewhere high or low, whether or not anyone has thought to look at it. The chapters ahead work out what it is made of: the conditions that raise it, the ones that quietly lower it, and what each of them looks like from the inside. The same

mechanisms keep surfacing in chapters that appear to be about quite different things, which is the clearest sign that this is one system rather than a set of separate subjects.

Why it feels more pressing now

Two things have shifted.

Building software used to be the constraint — the thing everything else queued behind. It gets cheaper and quicker every year, and as that friction falls away what is left is the organizational kind: who decides, who owns the outcome, whether anybody finds out if it worked. The bottleneck is moving, and it is moving somewhere most organizations have spent far less time thinking about.

The other is the word transformation, which usually arrives attached to a programme with a start date, an end date and a completion report. Run that way it treats the organization as a machine to be rebuilt and handed back, and it manufactures friction while it does: a programme comes with its own boundaries, its own steering group, its own reporting line and a handover at the end — most of the shapes in the next chapter, assembled on purpose. Meanwhile the thing it was meant to change carries on being what it always was, something living that never quite finishes, and the friction returns the moment attention moves elsewhere. That makes this closer to tending than to transforming — less a project to be completed than part of how an organization stays alive.

The other side of it

I have also seen the opposite, usually in organizations that looked no better resourced than anyone else. The same work moves more easily. Decisions get made, things get built, and

what comes out the other end reaches someone who actually wanted it. Nobody seems to be in much of a hurry. There is simply less in the way.

None of them got there by hiring more people, spending more money or buying better technology. That is the part I found interesting enough to keep pulling at, and it is what the rest of this paper is for: learning to see where the friction sits, and then finding the ways past it.

Fair warning: it is one of those ideas that is hard to unsee. Once you have words for it you start noticing it in meetings, in funding rounds, in org charts and in the gaps between them. You have almost certainly run into it already. You may just not have had a name for it.

The Cost of Value Never Created

Organizations rarely lose capacity. They lose the value that capacity could have created.

The finance team wanted printouts.

Not out of stubbornness. They needed to check the numbers, and checking meant reading them on paper, line by line, against another set of numbers, by hand. That was the control. It had always been the control, and it worked, in the sense that errors were caught.

It also meant that every month a group of capable people spent days doing something a machine could do in seconds. Everyone involved knew it. It carried on anyway, because the people who understood what the checks were *for* and the people who could automate them sat in different parts of the organization, speaking to each other through requirements documents.

So we tried something small. One person from finance came and sat close to the team building the thing — not as a stakeholder to be consulted at milestones, but close enough to be asked a question and answer it the same hour.

The printouts went away. Most of the manual validation became automatic. A bonus came with it that nobody had asked for: quality went *up*, not because the machine was more careful than the people were, but because once the checks were automatic we could afford to run far more of them than anyone would have done by hand.

Nobody in that story was doing anything wrong. Finance was protecting the numbers, which is their job. The team was building what was specified, which is theirs — and *what was specified* is where the whole problem was hiding, because a

specification is what you need when the people who understand the work aren't in the room. The friction wasn't inside the work. It was between the work.

One person moved, and thousands of hours went away. It is worth being exact about what that means, because "saved hours" is the phrase people reach for and it is the wrong one. Nobody was let go and nobody worked harder. The hours went away because the work that consumed them no longer had to be done by anyone.

What those people did next is the actual point. They went back to the work only they could do — the judgement calls about what the numbers meant, the questions nobody had been free to ask, the analysis that had been waiting behind a stack of printouts for years. Same salaries, same people, same week. A different amount of value coming out of the far end.

That is what structural friction costs. Not effort, not salaries, not even time exactly. **It costs the value those hours would have created if they had been spent on anything else.**

There is a team building software in that story, which makes it easy to file the whole thing as a technology problem. It isn't, and the examples that convinced me had no technology in them anywhere.

A recruitment team I worked alongside was measured on time to hire, and they were good at it. Roles that used to sit open for four months were closing in six weeks. Everyone was pleased, and reasonably so.

What nobody was measuring was what happened after someone said yes. The people arriving were a good fit on paper and knew very little about why the work mattered — partly because the slower conversations that would have told them were exactly what had been compressed out to save those weeks. Their new teams spent the first months filling that in, mostly in corridors, mostly without noticing they were doing it.

Six weeks was real and got reported. The months afterwards were just as real and appeared in nobody's numbers, because they arrived as a team that was quietly slower than it looked rather than as a cost of recruiting.

Nobody was wrong here either. Recruitment optimized what it owned and hit the target it had been given. The teams owned the consequences and not the decision. The friction was the distance between those two facts, and there was nothing to buy, nothing to build and no technology anywhere in the problem.

What I find striking about both stories is that you could interview every person involved and none of them would describe a problem. Ask each one whether they are doing their job well and they would say yes, and they would be right. The loss is real all the same, and it is sitting in the gaps between them, which means it belongs to nobody and removing it is nobody's job.

That is also why it stays hidden for years. Failure announces itself — a project cancelled, a system down, a customer lost. Friction does none of that. Projects continue, meetings continue, people stay busy, reports get delivered, approvals happen. Everything looks productive. The organization simply creates less than it could have, with exactly the same people, and there is no moment where anyone could reasonably have pulled a cord.

Friction turns up in a small number of recognizable shapes. I have found four worth separating, mostly because they fail differently and need different things done about them. They come back throughout the paper, so they are worth a minute here.

The first is **ownership nobody can say out loud** — where you can find plenty of people involved and nobody whose it is.

Ownership nobody can name. Two teams given the same capability on different time horizons — one building the future, one maintaining the past. It looks efficient on an org chart. In

practice both teams are technically right, neither is fully responsible, and every decision between them escalates.

Governance grown to fill the gap. Where ownership is unclear, people don't stop working — they start compensating. Meetings. Committees. Approval chains. None of it comes from incompetence. It comes from uncertainty, and uncertainty always builds something in its own defense.

The second is **context rebuilt at every boundary**, which is what a handover actually costs when you watch it closely.

Handovers, meaning any point where work passes from one group to another. Each looks like a clean transfer on a process diagram. Each is really a small act of forgetting, paid for later by whoever has to reconstruct what the previous team already knew. Sales to delivery. Recruitment to onboarding. Procurement to the department that has to live with what was procured.

The third is **the gap between a decision becoming necessary and being made**, which is almost never about anyone deciding slowly.

A decision waiting on a forum. The decision itself takes four minutes. Reaching the forum authorized to make it takes three weeks, and nobody counts the three weeks, because nobody bills for waiting.

Information that arrives correct and late. The report was accurate. It was thorough. It reached the person with authority to act on it after the window to act had closed. This is the most expensive form of friction and the least likely to be treated as a problem, because everyone involved did their job well.

A rhythm mismatch. The work needs a decision this week. The structure around it can produce one in April. Nobody is being obstructive; the two things simply run at different speeds, and the slower one sets the pace.

And the fourth is **what happens after**, or rather what does not: the outcome that never travels back to whoever chose.

Nothing changed as a result. Execution finished, results arrived, and nothing about the next decision was different. The organization didn't learn, so it will solve this problem again, at full price.

None of those are technology problems, though several will be presented to you as technology problems, which is its own kind of friction. Each is cumulative and invisible at once: no single instance is worth escalating, which is exactly why the total never gets escalated either.

If you want somewhere to start looking, count how many times a single piece of work changes hands before it reaches anyone outside the organization. The number is usually higher than people expect, and less interesting than how long it takes to establish it.

Playing with the numbers

What follows is not research. I have not measured this across a sample of companies and, as far as I know, neither has anyone else. What the numbers are good for is showing the shape of the thing — how something that never looks like much adds up to a great deal. Put your own figures in. Halve mine if you like; the conclusion is stubborn.

The part that still catches me out is how dull the inputs have to be. Nothing here needs a villain, a crisis or even a particularly badly run company. Four hours and three weeks are enough.

Start with the easy one. Ask a receiving team how long it takes before they can act without going back to ask. Four hours is a common answer and a modest one. Take a boundary crossed weekly:

> **4 hours × 52 weeks = 208 hours a year, on one boundary.**

More than five working weeks, spent reconstructing something somebody already knew. A mid-sized organization has eight or ten such boundaries without trying:

> **208 × 8 = 1,664 hours, or roughly a full-time role, doing nothing but remembering.**

Nobody is employed to do that. It appears in no budget and has never been justified to anyone, because at four hours a time it does not look like anything.

Waiting is the more expensive one, and it hides better, because the people waiting are not idle — they move to something else, which is precisely why nobody counts it. Take an initiative worth €50,000 a month once it is live, modest for anything a company bothers to fund at all. It needs eight cross-team decisions, made in sequence, each waiting a median of three weeks:

> **8 decisions × 3 weeks = 24 weeks of waiting, about six months.**
>
> **6 months × €50,000 = €300,000, in delay alone.**

Total deliberation across those eight decisions: perhaps a day. The organization did not lose six months of effort. **It lost six months of value creation.**

Nobody approves a six-month delay. Everybody approves *this particular* three-week wait, because this one has a good reason — and so does the next one. That is how a cost this size gets incurred without anyone ever deciding to incur it.

It is also why friction survives being obvious. It rarely shows up where it started. It shows up as slower delivery, more meetings, more approvals, more governance, more dependencies — symptoms, and often quite a reasonable response to uncertainty. Which is why removing a governance forum without settling the ownership question underneath it rarely creates speed. The symptom goes. The cause stays.

Faster is only half of it

There is a reading of all this that stops too early: clear the friction, and things move quicker.

That is true, and it is half the story. Moving quicker only helps if what comes out the other end is worth having. An organization can become extremely efficient at producing more of something nobody particularly wanted, and friction is just as much in the way of finding that out — the feedback that never travels back, the person who would have said *not like that* sitting three handovers away, the outcome nobody checks because the project was already marked delivered.

So there are two things to work on and they need each other. Producing more of what we already know how to produce. And getting more real value out of what we produce — value as the person on the receiving end actually experiences it, rather than as it appears on our own dashboard. The same friction sits in the way of both, which is the good news, because the same work moves both.

What comes back for the trouble is not only speed, though speed is what gets noticed first. The more useful change is that you end up watching both halves: how easily the work moves, and whether what came out of it actually landed with anyone.

Which is worth being concrete about, because it decides what you count. The four shapes above all measure movement — ownership, handovers, waiting, learning. Not one of

them can tell you whether the thing that moved was worth moving. That takes a fifth measure, sitting on its own and pointing outward: did anybody end up better off. Most organizations have some version of the first four, in one form or another, and have never quite got round to the fifth.

Neither half is something you install. Both have to be cultivated, and I spent years doing only that — working on culture and confidence while a structural problem sat untouched underneath. You do not cultivate your way out of one capability split across two departments; that one you fix with a decision. But once it is fixed, nothing improves on its own either.

Organizations rarely need dramatically more capacity. **They need more of the capacity they already have to actually become value** — and then they need to find out whether it did.

EDITOR'S NOTES

Naming waste and hunting it is not new. Lean has called it muda for decades, and the Theory of Constraints (Goldratt, The Goal) made the case that a system's throughput is set by one binding constraint rather than by total effort — which is the same argument as this chapter's, arrived at from manufacturing rather than from architecture. What is different here is the subject: Lean and TOC follow the work, and this paper follows the decision. Flow efficiency, the ratio of active time to elapsed time, is the Lean measure that most closely mirrors what these four indicators are reaching for.

What is Organizational Flow?

The movement itself — and what it looks like when nothing is in the way.

Every field has a word that everyone uses and nobody defines the same way. In this one it's *transformation*. Ask ten people what digital transformation means and you'll get ten answers, and the gap between those answers is where most transformation effort quietly disappears — not from lack of effort, but from lack of a shared destination.

So it's worth being precise once, because the rest of the paper leans on the distinction.

Digitization is the shallowest layer: analog into digital, paper into records. *Digitalization* goes deeper: better tools for the same work, done faster and with less friction, but still recognizably the same work. *Digital transformation* goes deeper still — rethinking not just how the work gets done but what's being done and why. And beyond that sits the rare case where an organization changes what it fundamentally is.

Most organizations aim for the third and settle for the second, and don't notice the difference until years in, when the tools have all changed and nothing else has.

Organizational Flow is not a fifth rung on that ladder. It's the ground underneath all four of them — the property an organization needs regardless of which rung it's currently attempting: the organization's ability to continuously move information, decisions, execution and learning through enduring capabilities, so that value gets created for customers and users.

Which gives the claim I would defend hardest in this paper. Digital transformation rarely fails for want of technology, budget or ambition — I have watched programmes with all three produce remarkably little. It fails when the organization underneath cannot get a decision to the person who knows the answer, cannot say who owns the outcome, and never finds out whether any of it landed. Put a transformation on top of that and you get new tools running old friction, faster.

So flow is not an alternative to transformation, and it is not a prerequisite to be finished first and then ticked off. It is what decides whether the transformation takes. Where it is working, an organization can aim at the third rung and reach it. Where it isn't, the same organization will do an enormous amount of good and expensive work and arrive, some years later, at the second one — with everything changed except what it is able to do.

There's a shorter form of that same claim, and it is the one to keep if only one sentence survives:

> **Organizational Flow rises as information, decisions, execution and learning move through enduring capabilities with clear ownership — and falls as structural friction accumulates faster than value is created.**

Read it slowly. Every word is load-bearing. Every part of this paper works on one element of that sentence. Information must move to the decision it serves. Decisions must move to where the knowledge and the accountability already sit. Execution must move, turning intent into something someone outside the organization experiences. Learning must move back, or the same mistake gets paid for twice. Enduring capabilities are what all of it travels through. And clear ownership is what makes any of it possible, because

someone has to be able to decide without asking. Remove any one element and the sentence stops describing an organization that moves.

Notice what does, and does not, move. Information moves. Decisions move. Execution moves. Learning moves. Value does not move. **Value is what gets created when the others do.**

Flow is a property, not a state

One thing about that sentence needs saying before anything is built on it, because getting it wrong turns a useful idea into a slogan.

Every organization has Organizational Flow. Not some. All of them, including the slowest one you have ever worked in. What differs is the level, and the level is set by one relationship: how much structural friction sits in the way, against how much value is being created despite it.

So flow is not a thing an organization acquires, arrives at, or completes. It is a property it holds a position on, the way a building holds a temperature. Positions move. They move in both directions, they move without anyone deciding, and they can be measured well enough to argue about.

This matters commercially more than it looks. An organization told it *lacks* flow hears a verdict and gets defensive, because the sentence is an insult with a diagram attached. An organization told it is *currently operating at a level of flow that costs it four months a year* hears a number, and numbers can be argued with, tested, and improved. The first framing starts a debate about competence. The second starts a conversation about structure — which is the only one worth having, because structure is the part anyone can actually change.

It also means there is no finish line, and nobody has to pretend there is one. You are not implementing flow. You are moving a position, and then holding it, because friction accumulates whether or not anyone is watching.

Two halves, and most organizations work on one

There is a second distinction inside that definition, and it is the one that decides whether any of this pays for itself.

Reducing friction makes the organization produce *more*. More output for the same headcount, arriving sooner, with less waste in between and less of everyone's week spent coordinating. It is the half that is easy to measure and easy to sell, because the before and after are both visible.

Creating value makes the organization produce the *right things*. Outcomes someone outside the building would miss if they stopped.

These are not the same, and neither one covers for the other. An organization can get very good at the first while quietly failing at the second, and the failure is invisible for a long time — because everything looks healthy. Throughput is up. Cycle times are down. The dashboards are green. The organization has become extremely efficient at building something nobody needed, which is the most expensive way there is to be busy.

The reverse is rarer and gentler: an organization that knows exactly what is worth building and takes three years to ship it. That one at least fails honestly, and usually loses to someone who could do both.

None of which is an argument against speed, and it would be a poor reading of this paper to take it as one.

Speed is good. I want to be unambiguous about that, because a paper full of warnings about producing the wrong thing quickly can start to sound like a case for going carefully, and it is not.

The speed that matters most is the speed of the loop rather than the speed of the work: how quickly you can try something, find out, and try again. And in almost every situation I have watched, the speed of iteration beats the quality of iteration. A team that ships something adequate on Tuesday and knows by Friday whether it helped will end the quarter ahead of a team that spent the quarter making the first version excellent. Not because care is bad, but because the second team is still guessing at the end of it, and the first has stopped.

The trap is narrower than "go slower". It is that iterating quickly only compounds if something comes back. Fast production with no return leg is not iteration at all — it is just producing, at speed, in whatever direction you happened to be pointing.

So both halves want speed. Reducing friction makes the loop turn faster. Creating value is what makes the turning worth anything.

Most organizations optimize output, because output is legible from the inside. Fewer optimize value, because value is only legible from the outside, and getting at it means asking people who do not work for you. So the measurement that is easy gets done, the measurement that matters gets postponed, and the organization slowly comes to believe that being busy and being useful are the same thing.

Both halves are required, and both can be cultivated to greatness — they simply need different attention. Friction is cultivated by subtraction: things taken out of the way of something that already wants to happen. Value is cultivated by

contact: with the person on the other end, with the outcome rather than the output, with evidence rather than the assumption written down at funding.

This paper works on both. Where a chapter is doing one rather than the other, it says so at the top.

Back to the long form, then, because the qualifiers in it matter too.

Continuously — not a project with an end date, but a condition. *Move* — flow is motion, not position; an organization can have excellent people, excellent technology and excellent intentions all rooted perfectly in place, growing nothing. *Information, decisions, execution, learning* — the four things that actually have to travel for anything to happen, and to go on happening. *Enduring capabilities* — the *what* that outlives every *how*. And *value gets created* — which is the part worth being precise about, because value is not one of the things that moves. Information moves. Decisions move. Execution moves. Learning moves. Value is what comes out the far end when all four of them do.

Here's why flow is worth naming separately from the ladder. Digitization, digitalization, transformation — each is something you *build*, and things you build can be finished and left. Flow is something you *cultivate*, and cultivation is never finished.

You can lay every irrigation line and still watch the crop fail, because nothing moved through them at the moment it was needed. The infrastructure was never the point. The growing was.

It's worth saying plainly what this looks like when it works, because the rest of this paper spends a long time on what stops it, and a reader could finish it never having pictured the thing itself.

Flow, in an organization that has it, is almost boring to watch. A decision gets made by the person standing closest to the problem, in the meeting where the problem came up, without anyone checking whether they were allowed to. A report reaches someone before the question it answers has stopped mattering. Two teams hand something to each other and neither one has to explain what the other already understood. Nobody is waiting on anyone, and nobody notices that, because there's nothing to notice — the absence of a wait doesn't announce itself the way the wait itself does.

It doesn't look like speed. It looks like the complete absence of a certain kind of friction, which is a strange thing to try to picture, because you're picturing something by its not being there.

What you get if this works

Six things, in rough order of how quickly they show up.

Speed you did not have to buy. Most organizations try to move faster by adding capacity. Removing friction is the only way to go faster that costs nothing and compounds — you are not adding energy to the system, you are stopping the system from consuming it. And capacity added to an organization with friction in it does not arrive as capacity. It arrives as more work crossing the same boundaries and more decisions queuing at the same forum.

Decisions made where the context is. When ownership is unmistakable, the person closest to the problem decides. That is not only faster; it is usually a better decision, made by someone who will still be there to live with it.

Governance you can actually remove. Committees created to compensate for unclear ownership become redundant the moment ownership is clear. This is the rarest thing in organizational change: a genuine subtraction, with no replacement required.

Learning that compounds instead of resetting. An organization that closes the loop pays for each mistake once.

People who stay. Capable people rarely leave because the work was hard. They leave because it was pointlessly hard, they could see exactly where it was being made harder, and nothing changed when they said so.

Adaptability, which is the only one that ultimately matters. Every organization eventually faces something it did not plan for. What determines the outcome is not the strategy. It is how fast the organization can move once it knows.

That is the definition, and it is the last piece of scaffolding this paper needs before it can be built on.

Which closes Part I. The friction has a name, the cost has a number, and the property being lost has a definition precise enough to argue with.

Part II sets out the enduring structures that decide whether an organization moves at all — purpose, ownership, capabilities, teams, people, information, decisions, execution, learning, and the value that comes out of the far end. **Part III** is about deliberately designing conditions with less friction in them. **Part IV** is where it leaves the page: how to observe friction, how to reduce it, how to create value rather than only more output, where to begin, and how this gets applied badly.

Along the way you'll find short essays in the margins — Insights, tagged to the chapter they belong to — and the occasional model, drawn closer to a tree than to boxes and arrows.

A few principles recur often enough to state once:

- **Technology is rarely the bottleneck. Friction is.**
- **Ownership before governance.** Governance is mostly what grows in the space where ownership is missing.

- **Capabilities endure.** Systems, suppliers and projects are how a capability is delivered this decade.
- **Capability belongs to the organization. Competence belongs to people.**
- **Information should move**, not accumulate.
- **Decisions belong where the knowledge and the accountability meet.**
- **Organizations are living systems**, not machines with a fixed working life.
- **Technology enables capabilities. It does not define them.**
- **People own. AI amplifies.** A system can execute a decision; it cannot be accountable for having made one.

The individual ideas are not new, and they are credited where they appear. Capabilities come out of enterprise architecture. Autonomy, mastery and purpose come out of motivation research. Psychological safety is Amy Edmondson's. The idea that organizations behave like living systems has a lineage running back decades.

What is new is bringing them together as a working theory of how organizations create, lose and regain Organizational Flow — with a vocabulary and a model built to make that theory usable. Everything claimed here is argued for, and qualified, in the chapters that follow. Where the evidence is thin I have said so there rather than here.

What is still missing is the thing itself. If flow moves through an organization, it moves through something — and that something has parts, each of which can be built well or badly, mostly by people who never knew they were building it.

We Stopped Building Machines

We learned to treat systems as living things, one deployment at a time, and went on treating the organizations around them as machinery.

Anyone who built software in 2001 will recognize the vocabulary.

Change requests. Change boards. Release windows. A maintenance budget kept carefully separate from a development budget, because one of them was for building the thing and the other was for keeping a finished thing running. A specification signed off before anyone wrote a line, on the reasonable theory that you would want to know what you were getting.

None of that was stupid. It was a coherent way of working, and it followed from one belief that everybody held and nobody examined: that a system is a thing you build, finish, and then look after.

That vocabulary is nearly gone now, and it went quietly. Nobody held a meeting to retire it. There was no memo announcing that the change board was over. The work simply stopped matching the words, and the words fell away — first in a few teams, then in most, until a graduate could join and never hear the phrase “release window” at all.

What replaced it took about fifteen years and arrived without ceremony. Continuous delivery. Teams that own a thing rather than hand it over. The quiet disappearance of “done” as a meaningful state. We stopped talking about finished systems because we stopped having any.

I lived through that shift from inside it, and the interesting part is how little of it felt like a decision. Almost nobody set out to adopt a philosophy. People just kept running into the same wall — the specification was wrong by the time it was signed, the handover lost something expensive, the finished system needed changing in month two — and worked around it, one deployment at a time, until the workarounds became the way things were done.

We learned, in other words, to treat our systems as living things. By instinct. Without ever writing down what a living thing actually is.

And then we carried on treating the organizations around them exactly as before.

The same people who would laugh at a two-year specification for a system will approve a two-year transformation programme with a completion date. The same organization that abandoned the change board still routes decisions to a monthly forum.

You can hold both beliefs for years without noticing, because they never meet in the same conversation. One is how you build. The other is how you are managed.

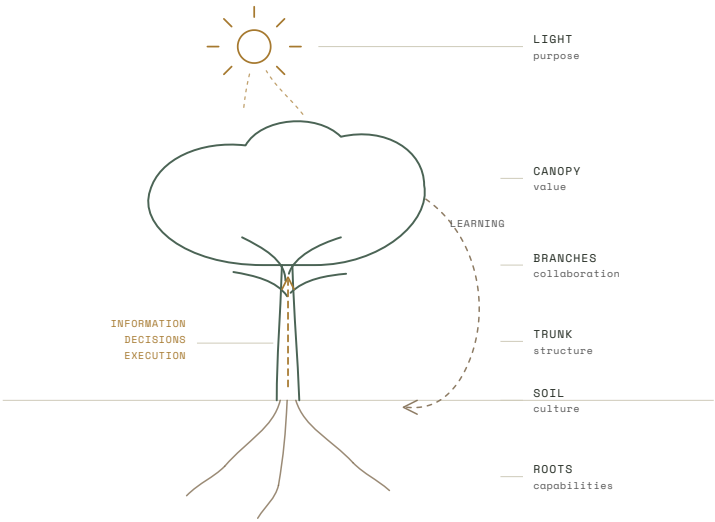
They do meet in the friction between them, constantly. A team that can deploy in an hour, waiting three weeks for a decision. A capability that evolves continuously, funded in annual chunks against a fixed scope.

If one thing is designed for continuous evolution and the other for completion, that mismatch is not an unfortunate side effect. It is where the friction comes from.

Which is the good news, oddly. We have already done this once, on the harder half, without a programme or a mandate — a lot of people working around the same wall until the workarounds became the way things were done. There is no reason the other half should be beyond us.

The Picture I Keep Coming Back To

One tree, standing in a forest of others. Every chapter after this is the same drawing from a different distance.



THE FLOW MODEL — LAYERS, AND THE CIRCULATION BETWEEN THEM

Here is the picture this paper uses, and it is the only one it uses.

I have tried others. Machines, pipelines, engines, orchestras, the human body. Every one of them was useful for about a chapter and then started lying — a pipeline has no roots, an orchestra has a conductor, and a body cannot be reorganized

in April. The tree kept working, and it kept working in the awkward places, which is the only real test a metaphor has to pass.

Light is purpose — the one part of this that isn't inside the tree at all, and the part most easily misread. A tree is not trying to reach the sun. It is trying to keep living: to grow, to adapt, to survive a bad season and come back larger. Light is one of the conditions that makes that possible, alongside water, soil, and room for roots — and what it does is orient the growing rather than serve as its destination. Purpose works the same way for an organization. Not a value stated on a wall, and not a finish line either, but the thing that decides which direction the growing goes in, whether anyone has named it or not. An organization without one does not stop. It grows in every direction at once, which is the expensive way of standing still.

Roots are capabilities — what the organization can always do. Invisible from above, doing the unglamorous work of holding everything else up.

Soil is culture — the medium every root grows through. Nobody sees it directly, and it determines almost everything about what's able to grow at all.

The trunk is structure — the load-bearing shape that channels what the roots gather upward. Thick enough not to bend in normal weather, and still, underneath the bark, alive and changing.

Branches are collaboration — the visible reaching-out between one part of the organization and the rest of it. Some load-bearing, some purely for light.

The canopy is value — the part the outside world actually sees. It looks like the whole tree from a distance. It's really just the most recent season's growth sitting on top of everything underneath.

And flow is the circulation — water and nutrients moving from roots through trunk to canopy, and back again as fallen leaves return to soil. The same way information, decisions and execution move through an organization, and the outcomes of that execution return as learning.

None of these layers work alone, and none of them work by being managed into place.

A machine responds to instruction. A living thing responds to conditions.

You cannot instruct a root system to grow faster; you can only improve the soil and wait. You cannot instruct a canopy to be more valuable; you can only strengthen what's underneath it and let the canopy be the honest result. This is the hardest habit to unlearn for anyone trained the other way — the impulse to intervene on the outcome rather than on the conditions that produce it.

A tree that stops circulating dies standing up, long before it looks dead from the outside. An organization where information doesn't reach decisions, where decisions don't reach execution, where execution's outcomes never make it back to inform the next decision, is doing exactly the same thing — just slower, and with better PR in the meantime.

Foresters had a name for this before organizations needed one. **Girdling** — a ring cut through the bark, all the way round the trunk, that stops the sap without felling the tree. The canopy stays green for weeks. A girdled tree doesn't announce that it's dying. It arrives as still being green.

That image is the reason I kept the metaphor. I have walked into organizations that were, in the strict sense, already girdled — still profitable, still shipping, still green from the road — and the only visible symptom was that nothing anyone decided seemed to reach anything.

Every chapter from here on is this drawing, looked at from a different distance. Not because the tree explains everything — no metaphor does — but because switching metaphors every chapter would be like redesigning the tree every season.

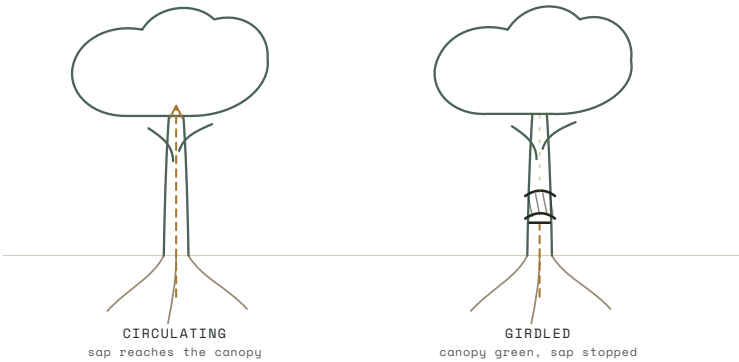
The value of a model isn't that it's perfect. It's that everyone can keep using the same one long enough for it to actually mean something.

One last thing about the drawing, because it decides how everything after it should be read.

The tree is *an* organization. Yours. Not organizations in general.

Around it stands a forest of others — competitors, partners, suppliers, the places your people worked before they came to you. Each is a tree in its own right, grown in its own soil, reaching for light that falls at a different angle, shaped by weather yours never had. Two organizations in the same industry, on the same street, are no more interchangeable than two trees on the same hillside.

Which is worth holding onto, because it is the quiet reason so much borrowed advice disappoints — including, quite possibly, some of mine.



Girdling — a ring through the bark stops the sap without felling the tree. Both canopies are green. Only one is still moving.

EDITOR'S NOTES

That organizations behave more like living systems than machines is not a new observation — Ludwig von Bertalanffy's general systems theory made the general case, and Donella Meadows' *Thinking in Systems* remains the most readable starting point. What this chapter does with it is more specific: it fixes one metaphor, assigns each layer of it to a named part of an organization, and then holds that mapping steady for the rest of the paper, so the metaphor can carry an argument instead of decorating one.

What a Capability Is, and Is Not

One word the rest of the paper leans on, defended once so it doesn't have to be re-explained in every chapter that uses it.

This is the shortest chapter in the paper and the one most likely to save you an argument.

One word recurs from here to the end: *capability*. It has been borrowed by enough people to mean almost anything, and a term that means anything decides nothing. So it is worth pinning down once, here, where anyone dropping into a later chapter can find it.

For this paper, a capability is an enduring business area — a bounded context that owns an outcome. Enduring, because it outlives the systems, suppliers and reorganizations that happen to deliver it this decade. Bounded, because you can say what is inside it and what is not. And it owns an *outcome*, not an activity: settling claims, rather than operating the claims system.

It answers exactly one question: what must we always be able to do? Not how, not where, not with which tool — those change constantly and always will. Onboard a customer. Price a policy. Ship a release. Resolve a complaint.

In the tree, capabilities are the root system. Unseen, unglamorous, and the reason anything above ground gets to grow at all.

Four things it keeps getting confused with

I have heard all four called capabilities in the same meeting, by people who were not being careless. The word invites it.

People. "We have a strong integration capability" almost always means five particular engineers. That is a competence, and it is a competence that can resign.

Resources. Headcount and budget are how much of something you have. A capability is a kind of thing, not a quantity of it. An organization can double the resources behind a capability nobody owns and get nothing but a larger queue.

Competence. This is the one worth slowing down on, because it costs the most: **capability belongs to the organization; competence belongs to people.** Underwriting is a competence — it arrives in the morning and leaves at night, and one day it does not come back. *Being able to underwrite* is a capability, and the organization needs it to survive that departure. The two are related in exactly one way: a capability is only as real as the competence currently staffing it. Which is why a capability with nobody behind it is an ambition on a slide, and competence with no capability to attach to is a talented person waiting to be told what they own.

Skills. These are the components competence is assembled from, and they sit a level or two below where anyone should be designing an organization.

The ten-second test

Ask *who owns this?* and you should get one name. Ask *who can do this?* and you should get several.

If the first question returns several names, or none, the capability has no owner. If the second returns exactly one, the capability is a person, and it will walk out with them.

I have run that test in a lot of rooms, and the useful part is not the answer. It is watching how long the pause is before the answer arrives.

None of this is about technology

Which is the part most often missed, probably because the vocabulary arrived through architecture.

A hospital must always be able to admit a patient. An insurer must always be able to settle a claim. A university must always be able to examine a student. Each of those has been performed by paper, by telephone, by three generations of software and by combinations of all three — and not one of them has changed as a capability in fifty years.

The methods are disposable. The capability is what the organization is left with when they go.

EDITOR'S NOTES

Thinking about what an organization must always be able to do is not new — it runs through enterprise architecture as a discipline rather than through any single method. The closest existing relative is Domain-Driven Design: Eric Evans' bounded contexts do much the same work of drawing lines around what belongs together and treating those lines, rather than the systems inside them, as the durable thing. Formal capability-mapping notations exist as well, and I have never found the notation to be the valuable part. What matters here is the capability as the unit of continuity an organization is designed around, which is a claim about organizational design rather than about modelling.

PART II

Seeing Organizational Flow

What have I seen again and again, in places that had nothing else in common?

Twenty-eight things I have watched repeat for twenty-five years. Observations rather than prescriptions — most of them were somebody's good idea first, which is exactly what makes them interesting.

Nothing here tells you what to do about any of it — that comes next. The aim is narrower and more useful: to describe each one clearly enough that you start spotting it yourself, in your own building, without me.

If a chapter makes you think "I have seen that", it has done its job.

CHAPTER 7 · PURPOSE

Busy Toward Nothing in Particular

Work carries on long after the reason for it quietly expired.

BOTH · WORKS ON OWNERSHIP CLARITY

Ask a team what they are working on and the answer comes back immediately, with detail, sometimes with a demo.

Ask the same team why, and something more interesting happens. Not embarrassment — everybody has an answer ready. But push one layer past it, to what the organization will be able to do afterwards that it cannot do today, and the room slows down. Somebody names a programme. Somebody else remembers a strategic priority from an off-site. There is usually a pause, and then the honest version arrives from whoever is least worried about their career: it was in the plan.

I have asked that question in a lot of rooms, and I have long since stopped hearing the vague answer as a sign that anything is wrong with the team. They are telling me the reason they were given. The reason they were given was about the work.

Which points at the thing worth noticing here: work does not stop when its reason stops. It has a budget, a plan, a team, a delivery date and a slot in somebody's reporting line, and not one of those is connected to whether it still matters. The reason was established once, usually at funding, in the single moment when the least evidence existed. Everything after that is execution, and execution is measured against the plan rather than against the reason.

So the work carries on, competently, past the point where anyone could say what it is for. Nobody notices, because there is nothing to notice. A thing that quietly stopped mat-

tering looks precisely like a thing that still matters, right up until someone asks.

This is the part I got wrong, and I got it wrong in the most standard way available. I treated purpose as something to communicate — write it clearly, say it often, make sure everyone has heard it. It produced recognition, which is not the same animal. People could recite the purpose back to me and still not tell me which decisions it made theirs. What I had actually given them was a sentence.

The version that held up was slower and considerably less tidy. It involved arguing about the boundary in the open, letting people push back until they could state the thing in their own words, and then not touching it again. That takes weeks and produces no artifact anyone can put on a slide, which is why it loses to the sentence most of the time.

There is an old parable about three stonecutters, which you have probably met before: one is cutting stone, one is earning a living, one is building a cathedral. Same work, different reason. It is a good story and it is half true, and the missing half is the half I care about. Give the third stonecutter no blueprint, no foreman and no reliable supply of stone, and his sense of building a cathedral survives about a fortnight. He is still just cutting stone, badly coordinated with whoever is cutting the next block.

Purpose held by one person is a good anecdote and a fragile arrangement — it lives in their head, and it leaves when they do, or when they have a bad month. Purpose that a group has argued its way into sits between them instead, which means you can lose somebody without losing the thread.

The effect on how an organization moves is quieter than you would expect. Purpose does not change what anyone does on a given morning. It changes what they reach for when they have two reasonable options and no time to check. Multiply that by a few hundred people making a few small choices

each, every day, for a year, and it is one of the largest forces acting on an organization while being entirely invisible in any given instance.

And when it is missing, there is no single failure to point at. Everyone is still busy, still reasonable, still working on something adjacent to what somebody else assumed they were working on. Which is why this one gets diagnosed as culture, or communication, or now and then as the wrong people — those at least are things you can name.

In the tree, purpose is the light. Not something the tree is trying to reach — it isn't going anywhere — but the thing that decides which way it grows. A root system isn't arbitrary either; it reaches toward water, and it does so long before anything is visible above ground.

Work does not stop when its reason stops, because nothing about a plan is attached to whether it still matters.

What moves this is not a better statement of purpose, which is the thing everyone reaches for and which produces recognition rather than ownership. It is having the argument in the open, letting a team push back on the boundary until they can state it in their own words, and then leaving it alone. That takes weeks and produces nothing anyone can put on a slide, which is exactly why the sentence usually wins.

Cultivating this: Leading Without Deciding, in Part III.

CHAPTER 8 · OWNERSHIP

A Meeting Grew Here

Where nobody owns a thing, people don't stop — they compensate, and the compensation becomes permanent.

BOTH · WORKS ON OWNERSHIP CLARITY

Every organization I have spent time in has at least one recurring meeting whose origin nobody can trace.

It is usually a coordination meeting. Two teams, forty-five minutes, every second week. Ask why it exists and you get a slightly awkward answer about alignment. Ask who set it up and you get the name of somebody who left in 2022. Nobody defends it and nobody cancels it, and on the one occasion somebody did cancel it, two things went wrong the following month and it came back stronger than before.

I do want those meetings gone. What changed is what I think they are for. They grew over something, the way scar tissue does, and what they grew over is usually the same thing. Somewhere upstream, two teams ended up looking after the same thing on different time horizons — one building the future of it, one keeping the present of it running. Nobody decided this; a reorganization moved a line, or a platform team was created, or a capability got split down the middle of two managers who both wanted it. On an org chart it looks efficient, even sensible. In practice both teams are right about their half, neither is responsible for the whole, and every question that falls between them has nowhere to go.

So somebody, quite reasonably, sets up a meeting.

And it works. That is the part I used to walk straight past on my way to the diagnosis. Questions that had nowhere to go now have somewhere to go, once a fortnight. Two teams who would otherwise learn about each other's plans afterwards

learn about them in advance. Something real is being held together in that room, which is exactly why cancelling it broke two things the following month.

What the meeting is doing, underneath all of that, is clearing uncertainty. Nobody in the room can say with any confidence who decides what, so every second week they establish it again by hand, for the next fortnight's worth of questions. That is a genuine purpose and it is genuinely being served.

It just costs those forty-five minutes permanently, from people who mostly cannot tell you why they are there. And because it works well enough, the uncertainty underneath it never becomes urgent enough to settle. The meeting takes exactly enough pressure off to keep itself necessary.

That is the pattern this chapter is about, and it generalizes further than meetings. When people cannot tell who owns an outcome, they do not stop working. They compensate. They build committees, approval chains, escalation paths, a shared spreadsheet with an owner column that everybody fills in differently. None of it comes from incompetence. It comes from uncertainty, and uncertainty is remarkably good at building things in its own defence.

The compensations are also, individually, correct. That is what makes them so durable. Each one solved a real problem on the day it was introduced, and removing it without addressing the thing underneath just reopens the injury.

Ownership is one of those words organizations use constantly and mean six different things by. Sometimes it means being the person left holding the outcome. Sometimes it means authority — someone who gets to say yes. The version this paper means is narrower and more useful than either:

Ownership is knowing, without asking, what's yours to decide.

Which gives the shortest version of everything above, and the one line from all this that has followed me across twenty-five years:

Governance grows in exactly the space where ownership is unclear.

The organizations I have seen with the least of it were not disciplined about removing it. They had left it nothing to compensate for.

That kind of knowing doesn't come from a title. It comes from a small set of agreed principles a team can lean on without asking — guardrails, which is where Part III of this paper picks the thread back up in full.

Purpose decided where the roots grow. Ownership decides who tends which one — and a root with no one tending it does not fail dramatically. It simply stops thickening, in a part of the tree nobody is looking at.

In every version of this I've watched happen, ownership created progress when it was *taken*, not when it was assigned. Nobody waited for a memo to feel ready — they took it, usually before anyone above them had noticed it needed taking, and the progress showed up afterwards, as the result rather than the cause.

Nobody ever scheduled the meeting that grew here. It grew because a question had nowhere else to go.

What moves this is not removing the meeting, which is the first thing everyone tries and which reliably brings it back within a quarter. It is settling the question underneath it — naming one owner for the thing two teams are each half-holding, specifically enough that a stranger could tell you who it is. Once that is settled the compensations stop being load-bearing and can come down without anything falling over. Done in the other order, you have removed the scar and left what caused it.

Cultivating this: Settling the Few Things, in Part III.

EDITOR'S NOTES

Psychological safety as a concept belongs to Amy Edmondson, whose research (see *The Fearless Organization*) is where the evidence lives. The claim this chapter adds is directional: that safety is largely a product of clear boundaries rather than a separate cultural programme run alongside them — guardrails first, safety as the result.

Nobody Could Say Management Put Me Here

The one time I watched ownership arrive on its own, and what it took.

BOTH · WORKS ON OWNERSHIP CLARITY

I have seen high ownership happen properly once, in the way it is supposed to, and I have been trying to work out ever since how much of it was the method and how much was luck.

We were standing up a new product organization. The normal approach would have been to draw the teams, staff them, announce them, and then spend six months explaining the reasoning to people who had not been in the room. We had all done that before, and we knew what those six months cost — not in effort, which we were happy to spend, but in the fact that the explaining rarely worked. People do not come to own a structure by having it explained well.

So instead we put everyone in a room. Developers, business specialists, architects, the cross-functional roles that usually get allocated last and told where they are going. Then we stepped back.

No manager decided who belonged where. No org chart dictated the outcome. There was one rule: every product needed a team with all the capabilities required to succeed.

The whole thing took less than two days.

What happened inside those two days was the part I did not expect, though I had hoped for it.

People did not optimize for themselves. I had braced for that — for everyone gravitating to the interesting product, or to their friends, or away from the difficult legacy thing nobody wanted. Some of that happened at the edges. Mostly it did not.

What they optimized for was the products. They discussed dependencies. They balanced skills. They noticed gaps and argued about who should fill them. Somebody moved off the exciting new thing because the unglamorous one would fail without them, and said so out loud, in front of everyone, without being asked.

Nobody instructed any of that. They did it because the outcome now depended on them, and they could see it depending on them.

The energy in that room is hard to describe without sounding like a brochure. People laughing, challenging each other, helping each other, solving problems together — not because it was required, but because they were building something they believed in and had just been handed the pen.

Every team left knowing why it existed, what it owned, and how it depended on the others. And not one person could say *management put me here*, because management hadn't.

Ownership was not assigned. It emerged.

Looking back, I do not think self-selection was the active ingredient.

That is the part people want to take away, and it is the part least likely to transfer. Run the same exercise in an organization where the last three reorganizations were announced rather than discussed, and you will get a room full of people waiting to be told what the right answer is, followed by a quiet renegotiation afterwards by whoever has the most political capital.

What made it work was underneath. We trusted people to make good decisions, visibly, in a way that would have been embarrassing to walk back. They understood the purpose well enough to make trade-offs against it. And they built the organization together, which is a different thing entirely from being handed one and asked for feedback.

The self-selection was how those conditions expressed themselves. It was not what produced them.

Which points at something I have seen hold everywhere since, including in rooms nothing like that one.

People rarely resist responsibility. They resist responsibility for decisions they never had the opportunity to influence.

That distinction accounts for most of what gets called change resistance in my experience. The person pushing back on the new structure is rarely opposed to the structure. They are being asked to own the consequences of something that arrived finished, and they can already see the parts that will not work, and nobody asked them at the point when saying so would have been useful.

Ownership handed down is a job description. Ownership taken is a different substance entirely, and the only reliable way I know to produce it is to be in the room earlier than feels comfortable, with less of it decided than feels safe.

Ownership taken has a different quality to ownership accepted, and I have never found a way to hand somebody the second and get the first.

Cultivating this: Drawing the Line Around a Team, in Part III.

EDITOR'S NOTES

Self-selection as a practice appears in sociocracy, Holacracy and Open Space facilitation, and the room described here was not the first of its kind. What the chapter draws from it is narrower: that ownership could not be assigned into existence, only chosen into it — which is why this paper puts ownership ahead of governance rather than treating governance as the way to produce it.

A Tollbooth on a Road With a Bypass

I built a control that people avoided, which meant I had built something else.

REDUCES FRICTION · WORKS ON DECISION LATENCY

A note on the word, for anyone dropping in here. This paper uses ***architecture*** *in a wider sense than the technical one — the deliberate shaping of how an organization is put together, of which system design is one part. A reorganization is an architectural decision. So is a funding model, a team boundary and a reporting line. What a Capability Is, and Is Not in Part I sets out the vocabulary this rests on.*

This one is mine, and I would rather tell it than have it inferred.

Years ago we had an architecture board. I was part of setting it up, and the purpose was entirely sound: significant decisions should get a second pair of eyes before they become expensive. Anyone who has watched a quiet choice about a database turn into a four-year constraint will recognize the impulse. It was not empire-building. It was care.

It also had good people in it. That matters, because the usual story about governance is that it fills up with people who do not understand the work, and this one didn't. The reviews were often genuinely useful. When a decision came to us, it usually left better than it arrived.

The trouble was the *when*.

People stopped bringing decisions.

Not because they disagreed with being reviewed. Nobody ever said that. They stopped because getting on the agenda, preparing the material, waiting for the slot, and then surviving the conversation cost more than the decision was worth to them.

Which is an entirely rational calculation, and one I would have made myself. A two-week wait and an afternoon of preparation is a fine price for a decision that will shape the next five years. It is an absurd price for a decision that needs making on Thursday and could reasonably go either way.

So they routed around it. Some decisions got made quietly and never surfaced — not hidden exactly, just never escalated, because the effort of escalating was the whole problem. Others did not get made at all, and sat waiting for somebody with the appetite to go through the process, which is a slower and more expensive failure that leaves no trace.

The board had been created to reduce risk. It ended up slowing everything down while seeing less than it had before it existed.

A control that people avoid isn't a control. It's a tollbooth on a road with a bypass.

I was also part of taking it apart again.

Which is worth saying plainly, because it is easy to tell the first half of a story like this and let the second half stay implied. The boards and review gates came down once we could see what they actually cost, and dismantling them was more uncomfortable than building them had been. Building a forum makes you look responsible. Removing one makes you look like you have stopped caring about risk, and somebody will say so.

The conclusion I drew first was the wrong one.

For a while I thought the lesson was about scope: we had reviewed too much, and a lighter version would have worked. That is wrong, and it is the mistake I see most often when

other people tell me the same story. A lighter tollbooth is still a tollbooth.

The board was not the mistake. Making a *destination* out of something that needed to happen where the work was — that was the mistake, and it would have been the same mistake whatever the board had been called. Design authority, technical council, review forum. The name changes nothing.

Any capability an organization turns into a place decisions must travel to becomes friction, however good the people standing in it are. The moment it is somewhere you go rather than something you have, the queue is the product.

What we had built, in the terms An Interface Nobody Can Find uses, was a queue — and a slow one, in front of a capability that people needed constantly. The bypass was not defiance. It was people finding a shorter route to the same answer, which is what people do when the official route costs more than the thing is worth.

And it generalizes further than architecture, which is the uncomfortable part.

Legal. Security. Procurement. Data governance. Every one of those is a real competence that organizations reliably convert into a location, and every one of them then notices that people are going around them.

The people are not the problem. The bypass exists because the road has a toll on it, and traffic is doing what traffic does.

A reliable sign that a control has stopped working is that everybody says nice things about it and nobody uses it.

Cultivating this: Equipping, Not Reviewing, in Part III.

CHAPTER 11 · CAPABILITIES

What Survives the Reorg

The organization keeps re-learning what it can do, because it files that knowledge under systems and teams instead.

REDUCES FRICTION · WORKS ON OWNERSHIP CLARITY

Sit through a reorganization and watch what actually changes.

The boxes move. The names change — competence centres become value streams become domains become platforms, on roughly a five-year cycle, and each generation is faintly embarrassed by the last one. Reporting lines get redrawn. There is a stretch of about three months where nobody is entirely sure who to ask about anything, and people quietly keep using their old contacts.

Then look at what people are actually doing. Claims still get settled. Customers still get onboarded. Releases still ship. The work the organization exists to do turns out to be almost completely unmoved by the exercise, which is either reassuring or expensive depending on what the reorganization cost to run.

That gap, between what changed and what carried on regardless, is what this chapter is about. There is a layer of an organization that survives its own restructuring, survives its systems, and survives most of the people currently staffing it. Very few organizations have ever written it down, which is why they keep rediscovering it — at full price — every time somebody moves the boxes again.

I once helped turn that layer into a card game.

It was for a planning workshop, and it sounds playful, which was the point. Formal documentation rarely gets a room full of people from different silos arguing productively about priorities. A deck of cards on a table does.

What the game did was move the conversation from *which system owns this* to *what are we actually trying to be able to do*. That shift — from solutions to capabilities — is usually where the real work of a transformation begins.

Everything below leans on one word, and it is defined in Part I rather than here — What a Capability Is, and Is Not, including the four things it gets confused with and the ten-second test for whether you actually have one. The short form: an enduring business area that owns an outcome, and outlives whatever happens to deliver it this decade.

Organizations that can name their capabilities clearly tend to survive their own technology decisions. Organizations that architect around tools instead rebuild the same understanding from scratch every time the tooling changes — and then wonder why each transformation costs as much as the last one.

Long before a capability shows up in a roadmap, it starts as a model — a way of seeing the organization that isn't the org chart and isn't the system diagram. Making one is closer to cartography than to documentation: you're not describing every feature of the terrain, you're deciding which features matter enough to draw.

A capability map is one of the most useful models available, because it's the one artifact a business stakeholder and an engineer can point at and mean the same thing.

It is tempting, especially now, to let the conversation start with the tool, because the tool is concrete and available and somebody is already selling it. The organizations that resist that pull are not more disciplined than the rest. They simply have somewhere else to start from, and the ones that don't have no choice but to start with what is in front of them.

Where it goes wrong from the other direction is when the map itself starts attracting effort. I have watched a capability map become a project, with a maintainer and a review cycle, at which point it has quietly become a system of its own. The artifact was never the point. The argument it forced into the open was, and that argument tends to outlive the diagram by several years.

Naming a capability is only half the job, though. Someone still has to be able to run it — and that is where the map stops being a diagram and starts deciding how people spend their Tuesdays.

What survives the reorganization is the work itself, and almost nobody writes it down.

What moves this is naming the handful of things the organization must always be able to do, in business terms rather than system names, and then funding and organizing around those rather than around whatever is delivering them this decade. The map is not the point — the argument it forces into the open is, and that argument outlives the diagram by years. Expect the first attempt to produce a list that is too long and too technical; the second one is usually the real one.

Cultivating this: Funding What Doesn't End, in Part III. An organization's capabilities shift as slowly as anything in it does, but they do shift, and a map nobody tends drifts out of date exactly when a reorganization or a new system makes it most worth having — which is where Part III picks the thread back up, in the chapter about paying for the things that never finish.

CHAPTER 12 · TEAMS

The Team That Has to Ask

Everyone needed to build it is in the room. Nobody in the room is allowed to decide.

BOTH · WORKS ON HANDOFF LOSS

Here is a team I have met a great many times, in different industries and under different names.

They are good. That is worth saying first, because when things move slowly the team is usually the first thing anyone questions. They know their part of the system better than anybody else, they ship reliably, and given a clear answer on Monday they would have something working by Thursday.

What they do not have is the answer.

So the question goes out. To a product owner, who takes it to a stakeholder. Or to a forum that meets on Wednesdays. Or to a genuinely helpful person in the business who genuinely has four other things on. It comes back in a few days, occasionally a few weeks, sometimes subtly changed by the journey — because the person who eventually answered was not the person who originally asked, and something got smoothed over on the way.

Meanwhile the team works on something else, because they are professionals and there is always something else. Nobody logs the wait. On paper they are fully utilized, which is among the more misleading numbers an organization collects about itself.

Ask anyone involved and they will describe a perfectly functioning arrangement. There is a process for questions. It has an owner. It usually works. And the thing that was supposed

to take a week has taken a month, without a single person doing anything other than their job properly.

What that team has is everything needed to build, inside the boundary, and nothing needed to decide. The boundary was drawn around the work rather than around the judgement, and the two are not in the same place.

And there is a specific reason this keeps happening, which most of the advice about team design steps around.

Almost every model in this space was written for technology teams, and it shows in one way: the business perspective enters as a role rather than as authority. A product owner. A stakeholder. Someone who represents.

Representation is the problem, not the solution.

A team with a technical product owner has somebody who can prioritize, translate and negotiate, and who cannot answer the question that actually blocks the work: *should we do this at all, and what happens to the customer if we get it wrong?* That answer lives elsewhere, so the team asks, and waits, and the handover the boundary was drawn to remove reappears wearing a different job title.

It is worth being exact about what such a person can and cannot hold, in the terms Everything Was Green sets out. A technical product owner can own the **output** — the right thing gets built, correctly, in a sensible order, at a decent pace. That is real and it is not nothing.

What they cannot own is the **outcome**, because owning an outcome means being answerable for whether the person on the other end ended up better off, and that requires the standing to say *we should not build this* and be believed. A representative does not have that standing. It is not a competence problem and no amount of seniority in the role fixes it — the authority sits with whoever the representative is representing.

So the team has an owner for the half that was never in doubt, and none for the half that decides whether any of the work mattered.

A team built around a capability needs someone in it who can decide about that capability — not describe what the deciders want. Someone whose judgement about the domain is trusted, who carries the consequences, and who does not have to check.

The clearest version of this I've seen wasn't a technology team at all. A finance team wanted printouts — numbers checked by hand, line by line, against another set of numbers. It worked, in the sense that errors were caught, and it also meant a group of capable people spent days every month doing something a machine could do in seconds, because the people who understood what the checks were *for* and the people who could automate them were in different parts of the organization, talking through requirements documents.

One person from finance moved. Not as a stakeholder consulted at milestones — close enough to be asked a question and answer it the same hour. The printouts went away. Most of the manual validation became automatic. Quality went *up*, because once the checks were automatic the team could afford to run more of them than anyone would ever have done by hand.

Nothing else changed. No new technology. No new process. No reorganization. One handover disappeared, and the friction it had been generating, quietly, every week, went with it.

That is what "delivering *to* the business" actually costs. It sounds like good service. In practice it describes a structural boundary — a specification at one end, an acceptance at the other, and a gap in the middle where nobody truly owns whether the outcome solves the problem it was meant to solve. Both sides can perform perfectly and still deliver something nobody needed.

A team made up only of builders can build almost anything. But it cannot decide very much.

Which is why the teams that worked were never made only of builders.

The finance story above is one example, and the room in Nobody Could Say Management Put Me Here is the other. Neither produced a development team with a stakeholder attached. Both produced something we ended up calling a *business product* team — business people, developers and operations in the same team, building the thing and running it, and answerable for the outcome rather than for the handover.

The industry name closest to it is BizDevOps, and the name matters less than what it removes. DevOps closed the gap between building and running, which was a real and expensive gap. It left the older one untouched: the gap between deciding what is worth building and building it. A team that builds and operates but still receives its requirements has closed one seam and kept the one that costs more.

None of that is a bigger team. It is usually the same headcount with a different composition — one or two of the builders replaced by the people who would otherwise have been on the other end of the specification. Which is a harder conversation than adding a product owner, because somebody has to give up a person they consider too valuable to embed.

Staffing a team this way is harder than staffing it with a product owner, which is precisely why most organizations don't. It requires giving up a person the business considers too valuable to embed in one team. And it converts the most expensive recurring handover in the organization into a conversation across a desk, which is a good return for a decision that costs no headcount.

The test is simple enough to run this week. Take a team and ask what they had to escalate last month. If any of it was a question about the domain rather than about money or headcount, the team is arranged around the work but not around the decision.

A team you have to interrupt someone else to run is not a team. It's a queue with a name.

What moves this is not a better process for questions, which is what gets tried and which makes the queue tidier rather than shorter. It is moving one person — the one whose judgement the team keeps waiting on — inside the line, and then letting them decide without checking. That is a staffing decision rather than a process one, and it costs somebody a person they consider too valuable to embed. Nothing else I have tried converts a recurring handover into a conversation across a desk.

Cultivating this: Drawing the Line Around a Team, in Part III.

CHAPTER 13 · TEAM SIZE

How Much Can One Team Hold

Counting people tells you almost nothing. Counting how many separate things they must understand tells you most of it.

REDUCES FRICTION · WORKS ON HANDOFF LOSS

There is more advice available on how big a team should be than on almost anything else in organizational design, which is usually a sign that nobody has quite settled it.

Amazon's two-pizza rule sizes a team by how many people you can feed. Dunbar's numbers size it by how many relationships a person can hold in their head — and those numbers are considerably more contested than their popularity suggests, though the shape of the idea survives the argument. Agile orthodoxy lands on seven or eight and calls it experience, which is at least honest about where it came from.

All of them are approximating the same underlying constraint from different directions: a team should be small enough that everyone in it can hold the whole thing in their head. Not the whole organization. The whole of what they are responsible for.

Which is why headcount is the wrong number, and also the one everybody reaches for, myself included.

Team Topologies gives the constraint its most useful name — **cognitive load** — and it is a better sizing rule because it measures the thing that actually binds.

A team of five carrying four unrelated domains is overloaded. A team of nine carrying one coherent capability may be entirely comfortable. Counting people tells you almost

nothing; counting how many separate things they must understand tells you most of it.

I have watched an organization "fix" an overloaded team by adding two people, which made the load worse — more coordination, more context to share, and the same four unrelated domains still sitting there. Then watched a different one fix it by moving one domain out, with no change in headcount at all, after which the team was visibly faster within a month.

And it explains why the capability is the right thing to draw the boundary around.

A capability is, by definition, coherent — a single thing the organization must always be able to do. Draw the team there and the cognitive load has a natural edge; the things they must understand belong together, so understanding one helps with the next.

Draw it around a technology, a system or a reporting line, and the team inherits whatever collection of unrelated concerns happened to end up in that box. Nothing about them helps with anything else. That is the same nine people, carrying much more.

What happens between teams is a design choice too, and mostly gets made by accident.

Not every team is the same kind of team. Some deliver a stream of value directly. Some provide a platform others build on. Some exist to raise competence elsewhere and then get out of the way — and that last kind is the one organizations forget to disband, at which point it becomes a permanent department looking for something to do.

The ways teams interact are equally a choice rather than a personality outcome. Close collaboration is expensive and should generally be temporary. Consuming something as a service is cheap and should be the default once the thing is stable enough to have an interface at all.

In this paper's terms both of those are statements about friction. Collaboration is a handover you have chosen to keep, usually for a good reason, ideally for a stated period. A service is a handover you have removed.

The failure is rarely picking the wrong one. It is never noticing there was a choice — inheriting the interaction pattern from whatever the first integration happened to do, and then treating it as the nature of the relationship.

A team is not overloaded because it is small. It is overloaded because somebody drew a box around things that have nothing to do with each other.

Cultivating this: Drawing the Line Around a Team, in Part III.

EDITOR'S NOTES

Team types, interaction modes and cognitive load as a sizing constraint are Matthew Skelton and Manuel Pais's contribution in *Team Topologies*, and anyone designing team boundaries should read the original rather than this summary of it. The two-pizza rule is Amazon's; the relationship numbers are Robin Dunbar's, and are more contested than their popular use suggests. What this chapter adds is only the framing — that all of the sizing advice is approximating the same constraint, and that the constraint is about understanding rather than headcount.

CHAPTER 14 · PEOPLE

The Wait Nobody Logged

The queue that appears in no system, because the people in it moved on to something else.

CREATES VALUE · WORKS ON DECISION LATENCY

Somebody sent a message on Tuesday morning. A short one — a question they needed answered before they could carry on with anything.

The answer arrived Thursday afternoon.

Nothing went wrong there. The person who answered was not ignoring anyone; they were in workshops for two days, it was not marked urgent, and they replied properly as soon as they had a moment to think about it. Two days is a civil response time. I have waited considerably longer for less.

Two days is also two days. It appears in no system, on no report, in nobody's velocity, and if you asked either person about it a week later neither would remember it happened. Now multiply it by everyone who is currently waiting on somebody else, which is very nearly everyone, and you have what I suspect is the largest single category of delay in most organizations — and the only one with no number attached to it anywhere.

This is the chapter where the structural argument runs out. You cannot draw a boundary that makes people answer each other faster. What closes that gap is somebody deciding another person's problem is worth ten minutes of their afternoon, and there is no org chart that produces that.

That is the whole observation, and it is smaller and more stubborn than most things in this paper.

Waiting on another person is the most common form of friction in any organization and the only one with no instrument pointed at it. A queue in a system can be measured. A ticket has an age. But the question sitting in somebody's inbox behind four other things has no age, no owner, and no record that it was ever asked — and when it is finally answered, both parties experience the exchange as having gone fine.

Multiply it properly and the number stops being comfortable. If everyone in a two-hundred-person organization is waiting on somebody for an average of a day a week — which is a conservative guess and I would take a higher one — that is forty people's worth of time, permanently, spent on nothing, in a place where every individual instance is unremarkable and nobody is doing anything wrong.

In the vocabulary this paper uses elsewhere, every one of those waits is a queue — just one with no name, no owner and no visible position in it. Which is why it is the hardest kind to shorten: you cannot staff a queue nobody has admitted exists.

The reason it stays invisible is that the waiting is never anybody's whole day. People move on to the next thing, which is exactly the right professional response and exactly what removes the evidence. Nobody sits idle. The work just arrives later than it would have, and there is nothing to point at.

What moves this is not a system, and I have watched several organizations discover that expensively. It is closer to a norm — that a question from someone blocked gets ten minutes today rather than a proper answer next week, and that unblocking somebody counts as work rather than as an interruption to it. That is cultural rather than structural, which makes it slower to establish and much harder to lose once it holds.

The largest queue in most organizations is in other people's inboxes, and it has never appeared on a single report.

Cultivating this: Ten Minutes Today, in Part III.

Nobody Opts Out of the Arithmetic

Every principle in this paper still has to pass, at some point, through one person choosing to help another.

CREATES VALUE · WORKS ON DECISION LATENCY

One thing I have carried with me from sports is a high expectation of people. Not for perfection — for contribution.

On a team you depend on others and they depend on you, and nobody gets to opt out of that arithmetic just because they are having an off week. It is not a moral position. It is the shape of the thing. Eleven people on a pitch, and the one who stops running does not fail alone; they fail everyone standing in the space they were supposed to cover.

Organizations are no different, even though they dress the dependency up in job titles and org charts instead of positions on a pitch. People show up with different motivations, different ambitions and different strengths, and none of it matters nearly as much as whether, when the work depends on each other, everyone's contribution actually arrives.

Flow has a human dependency, and it is absolute.

Notice how many of the verbs in this paper require somebody to choose.

Information creates value when it is *shared*. Decisions create momentum when they are *made*. Ownership creates progress when it is *taken*.

Not one of those happens on its own. Every argument in these pages about capabilities and boundaries and where decisions should sit still passes, at some point, through one person deciding to help another person move — and no structure yet devised makes that decision for them.

Which is why I have stopped being surprised when two organizations with almost identical structures perform completely differently. The structure sets the ceiling. What happens underneath it is a few hundred small choices a day about whether somebody else's problem is worth ten minutes.

Contribution works the way a root system does, which is to say invisibly and in aggregate.

A single root does not feed a tree. What feeds it is the whole network of small, unglamorous transfers, none of which is remarkable on its own.

The organizational version is the answer given quickly, the blocker removed before anyone had to escalate it, the half-hour spent explaining something to somebody who could have been left to work it out. Every one of those ends a wait that nobody logged — which makes contribution the only force in this paper that directly removes the only friction with no number attached to it.

Trust is built out of the same material. It does not grow from stated values, and it does not arrive at an offsite. It grows when people consistently help each other move: the same small transaction repeated often enough that it stops being a pleasant surprise and becomes a reasonable expectation. That is a slow accumulation and a fast loss, in that order.

And noticing is the part that gets called soft, which is a mistake.

Noticing that somebody is waiting — actually noticing, not acknowledging it in a retrospective — is what empathy looks like at work. It is not a personal quality kept separate from the real work. It is the specific attention that makes the real work move, and it is rarer than it sounds, because a person waiting quietly produces no signal at all.

Every organization looks different from the outside. Different products, different structures, different vocabularies. The interdependence underneath is not different anywhere. People

are still waiting on other people, and how well that waiting resolves is most of what the word *culture* is actually measuring.

Flow is not something an individual can produce alone, however good they are. It comes out of what people are willing to do for each other.

Cultivating this: Ten Minutes Today, in Part III.

CHAPTER 16 · TALENT

Downstream of the Conditions

Attracting the people who can build the distinctive part is almost never a recruiting problem.

CREATES VALUE · WORKS ON ALL FOUR

Somewhere in most organizations there is a slide about the war for talent, and somewhere else there is a team that has been trying to fill the same two roles since March.

The two are usually treated as the same problem, handled by the same function, and solved with the same instruments: a better advert, a wider net, a recruitment partner, an employer branding exercise. Sometimes that works. Often it produces a lot of activity and the same two open roles in September.

There is a structural reason this matters more now, not less, and it starts a long way from recruiting.

More and more organizations create and defend their value through systems they build rather than systems they buy.

Not because they set out to become technology companies — most emphatically do not want to be, and say so — but because the things that distinguish a bank from another bank, or a retailer from another retailer, increasingly live in software nobody else has.

This is easy to say badly, so: it is not a claim that everyone should think like an engineer, or that the business exists to serve the technology. It is narrower than that. If the distinctive part of what you do is now built rather than bought, then the ability to build it well has quietly become one of your en-

during capabilities — and it will go on being treated as a support function on the org chart for years after that stops being true.

That gap, between what a capability has become and where the org chart still files it, is where a surprising amount of friction lives. A support function gets a budget and a service level. A distinguishing capability gets attention, ownership and the argument about what it should be. Very few organizations notice the moment one turned into the other.

Which changes what the word *people* means as a resource.

Software-driven organizations do not need more hands. They need people who can think in systems — who can hold a whole problem, see where it connects, and make a judgement about it — and those people have more choice about where they spend their time than almost any other kind of talent in the market.

Attracting them gets treated as a recruiting problem, and it rarely is one. The recruiting is downstream of whether the organization is somewhere those people would choose to be, and no amount of work on the funnel compensates for the answer being no. You can improve a process considerably and still lose every candidate at the point where they ask what a normal week looks like.

The tech stack does attract them. It is not what keeps them, and that is the part organizations get wrong.

Modern tools signal interesting problems, a place that invests in engineering, work that will still be relevant in three years. That gets people through the door, and dismissing it is a mistake I have watched people make.

What decides whether they stay is the environment around the stack — whether they can learn, influence a decision, do work that matters, and actually get things done. Which is mostly a question of transparency about the journey. The

tools, yes, but also the mistakes, the lessons, the actual texture of what it is like to work somewhere — told honestly rather than curated.

Technologists can tell the difference between an organization confident enough to talk about what went wrong and one only willing to talk about what went right, and they can tell in about four minutes. The second reads as marketing. The first reads as a place where you would be allowed to be wrong occasionally, which is the thing they are actually trying to find out.

So the honest version of the talent problem is uncomfortable, and it is the reason this chapter sits in the observations rather than in the advice: most of what determines whether you can hire the people you need was decided months earlier, by people who were not thinking about hiring at all. How decisions get made. Whether a team can own something. What happened to the last person who raised a problem.

Most of what a candidate is trying to work out in an interview, the organization already answered, in public, without noticing.

Cultivating this: Growing the Team, in Part III.

CHAPTER 17 · SYMPTOMS

It Shows Up in People First

Long before friction appears in a number, it appears in how people sound.

BOTH · WORKS ON ALL FOUR

The friction that does this to people mostly sits between teams rather than inside them. A team pushing on something that is not theirs to move — another team's queue, a boundary drawn somewhere else, a decision that lives two floors up and arrives when it arrives. It happens within a team as well, and that version tends to get sorted out, because everyone involved is in the same room and can see each other's week.

Where the thing being pushed on is outside the team, and it has been going on a while, you tend to find two people in that team. The uncomfortable part is that they're usually the same person, six months apart.

Start with the loud one, because they come first. They are frustrated, sometimes visibly angry, and they are the easiest person in the room to misread. The frustration is not apathy — it's the opposite. They still believe the work matters. They can see the thing clearly and cannot get to it, and that gap is exactly what frustration is. Nobody gets frustrated about work they've stopped caring about.

Now the quiet one. They still attend, still deliver what's asked, still say the right amount in the retro. But the suggestions stopped a while back, and nobody noticed the exact week it happened.

That's the same person, later. They pushed three times, got three reasonable explanations for why nothing could change, and drew the only sensible conclusion available: this is not a

thing I can move. What gets read as disengagement is usually just accurate learning. The energy didn't fail — it got spent, on a wall, and there was none left over.

So the sequence matters more than the two types do. Frustration is what caring sounds like while it still has somewhere to go. Silence is what's left afterwards.

Which means the loud phase is the one you want to catch, and it's the one organizations reliably mishandle.

A frustrated person is the last, best signal you get before somebody stops trying — and they usually get treated as a behavioural problem right at the moment they're being most useful. The tone gets noted. Somebody has a word about how it's coming across. And that conversation, however kindly meant, teaches exactly one lesson: this is not welcome here. Which reliably produces the quiet version, and everyone reports that things have settled down.

They have. That's the problem.

Frustration also costs more than silence while it lasts, and not for the reason people assume.

Frustration has to go somewhere. So it goes to a colleague at the coffee machine, then to two colleagues in a side conversation after standup, then into a message thread that pulls in three more people who now have opinions about it. Each of those conversations is somebody's attention, taken out of the work and spent on the reasons the work is difficult.

And here's the part that stings: those conversations are almost always held by the people who care most. The ones venting are venting *because* they wanted to create something and couldn't. The only thing they want is to do the work. The friction takes the work away from them and then takes their time again in the discussion about why.

That is friction generating more friction, and it compounds quietly. One blocked person becomes an hour of three people's afternoon. The same blockage next month becomes a

running theme. Six months in, it isn't a blockage anymore — it's a story the team tells about another team, and stories are much harder to remove than blockages ever were.

It spreads, because that's what shared attention does.

Nobody joins a team already cynical about the approval process. They learn it, in about a fortnight, from whoever is nearest. Not through anything formal — through tone. Through what gets sighed at. Through the small joke about how long a request will take, which everyone laughs at because everyone recognizes it.

The organization did not decide to teach that. It simply became the most accurate available description of how things work, and people transmit accurate descriptions to each other for free.

And it builds walls, which take far longer to remove than the friction that built them.

This is the expensive part. A structural problem — an unclear boundary, a decision sitting in the wrong place — can often be fixed in an afternoon once someone is willing to fix it. The wall it produced between two teams over eighteen months does not come down in an afternoon. It comes down over quarters, through repeated small experiences of the other side behaving differently than expected.

Fix the structure and the relationship does not automatically follow. You have removed the cause and inherited the residue, and the residue is now its own source of delay: the request phrased defensively, the meeting invited to as a precaution, the extra person copied in just in case.

None of this appears anywhere you're looking.

Task switching, the interrupted afternoon, the third rehearsal of the same complaint, the wall — none of it lands in a report. What lands in a report is that a team's output

dropped slightly, which gets read as a delivery problem, which invites exactly the kind of attention that makes it worse.

The signals are all in the room instead. Someone who used to push and stopped. Someone who is angrier than the situation seems to warrant. Meetings where the interesting conversation happens afterward, in twos, in the corridor. A team that has started explaining itself before anyone asked.

The people showing symptoms are not the problem. They are the earliest, cheapest, most honest reading you will ever get of a structure that is costing you something — and the reading gets worse the longer you leave it, because the loudest signal is the first one to go.

Frustration is the sound of someone still trying. Silence is the sound of someone who has stopped believing it will make a difference.

What moves this is noticing early enough that there is still something to notice, which is a leadership practice rather than a structural fix. The frustrated person has almost certainly identified something real and is the only one still willing to say it out loud; the quiet one will need asking more than once, because the honest answer is unflattering to whoever is asking. Both questions are cheap. The second is much harder to act on, which is the whole argument for asking the first one earlier.

Cultivating this: Growing the Team, in Part III.

CHAPTER 18 · ALIGNMENT

Everyone Left Agreeing

Nobody lied. Nobody misheard. Everyone simply left with a slightly different picture in their head.

BOTH · WORKS ON HANDOFF LOSS

The meeting ended a few minutes early, which everyone took as a good sign.

Eight people, an hour, a genuinely productive discussion. Nods at the right moments. No unresolved argument. Somebody said "great, I think we're aligned," and everyone agreed that yes, we're aligned, and went back to their desks.

Three weeks later it turned out that two of those eight had been building toward noticeably different things. Not opposite things — that would have been caught immediately. Adjacent things. Different enough to matter, similar enough that every status update along the way sounded fine.

Nobody was careless. That's what makes this one worth a chapter.

Agreement in a room is mostly a feeling, and the feeling is unreliable.

What actually happened in that hour was that eight people heard the same words and each constructed their own picture from them — filled in with their own context, their own assumptions about scope, their own sense of what "we'll handle that later" covers. Everyone left holding a picture. Everyone assumed it was the same picture, because the words had been the same.

Words are extremely low-bandwidth compared to the mental models they're standing in for. Most of the model stays in the head it came from, unspoken, because it's obvious — to the

person holding it.

The gap is invisible at exactly the moment it's cheapest to close.

This is the cruel timing of it. In the room, with everyone present, the correction costs one question and ninety seconds. Three weeks later it costs whatever was built in the meantime, plus the conversation about how we got here, plus the small dent in trust between two people who now each privately suspect the other wasn't listening.

The cost of the assumption is paid entirely in the future, which is precisely why nobody spends the ninety seconds in the present.

Nobody asks the checking question, for a very human reason.

Asking "can I say back what I think we just decided?" feels, in the moment, like admitting you weren't following. It slows a meeting that is going well. It risks looking slow in front of people whose opinion you care about. And everyone else is nodding, so presumably it's just you.

Except everyone in the room is running that same calculation. Which is how eight people who each had a small uncertainty all decide, independently and simultaneously, to keep it to themselves.

Handovers are the same failure with a longer fuse.

A handover is a meeting where half the participants aren't in the room. One side has a rich, detailed model built over months. The other side receives a document and a conversation. Both parties agree the handover happened — there was a session, there were slides, questions were invited and none seemed pressing.

What transferred was the vocabulary, not the model. The receiving side now has enough language to describe the thing and not enough understanding to make judgement calls

about it, and that gap only becomes visible when the first non-obvious decision arrives.

This is friction that generates no friction at all, right up until it generates all of it.

Nothing feels slow. Nobody is waiting. Everyone is productively building — which is what makes false consensus the most efficient way an organization has of moving quickly in slightly different directions.

And it is worth noticing how much of this paper it quietly touches. The handover that loses context, the decision that meant something different by the time it arrived, the team that built the adjacent thing. Several of the shapes elsewhere in these pages are this one, wearing different clothes.

The most expensive gaps are the ones where everyone was sure they agreed.

What moves this is cheap and slightly awkward: someone saying back, out loud, what they are going to go and do, while the people who could correct them are still in the room. Half the time it costs ninety seconds and nothing happens. The other half recovers three weeks. What makes it hard is not the technique but the social cost of going second — which means it works when whoever has the most standing in the room does it first, and unreliably otherwise.

Cultivating this: Reducing Friction, in Part III.

CHAPTER 19 · PLACE

Where the Work Actually Happens

The friction of place is personal, which is exactly why it keeps being solved collectively.

BOTH · WORKS ON HANDOFF LOSS

Covid was a tragedy and I have no wish to live through anything like it again.

It also settled an argument that had been going nowhere for a decade. Working from home went from something that required justification — occasionally something close to forbidden — to something an entire economy did on a Wednesday, because there was no alternative. Whatever else that period cost, it removed the need to speculate about whether it could work.

What we learned is more interesting than either camp wanted.

For a lot of people it removed a real and unglamorous friction.

Not work friction. Life friction. The commute, the school pickup that required a favour, the appointment that consumed a half-day of leave, the daily negotiation between being a functioning employee and a functioning family member. For many people that friction was substantial, invisible to their employer, and being paid for entirely out of their own reserves.

Removing it did not make those people lazier. It gave them back capacity they had been spending on logistics.

For others it was genuinely worse, and that's not a character flaw either.

Some people found the distractions unmanageable — the washing machine, the children, the house full of small unfinished tasks. Some people need other people in the room to think properly. Some found the isolation corrosive in a way that took months to admit to.

And some found, for the first time in their working lives, the uninterrupted focus they'd always needed and never had, and produced more than they ever had in an office. That happened too, quietly, to more people than the debate usually allows for.

All of these are true simultaneously, about different people, in the same team.

Which is why the return-to-office argument keeps missing.

The friction here is *personal*. It varies by individual, by task, by week, by what's happening at home this month. Any policy that resolves it in one direction resolves it correctly for some fraction of people and incorrectly for the rest, and no amount of conviction about collaboration changes that arithmetic.

I don't hold a position on where people should be, because I don't think there's a position to hold.

The argument I hear most is that people don't actually work at home. It's worth taking seriously, because it isn't entirely wrong.

Some people genuinely do less at home. That's true, and pretending otherwise makes the rest of the case sound naive.

It's also nowhere near the whole picture. Plenty of people work *more* at home — longer, later, and with far more difficulty stopping, because the thing that used to end the working day was a commute and now nothing does. That's a real risk too, and it's the one nobody writes a policy about, presumably because it looks like commitment.

And wherever that is happening, it was usually happening in the office too — just where somebody could see it, which an office is very good at and which is a different thing from it not happening. The building also changes the shape of it. Time that goes quietly at home goes sociably in an office: the long chat at a desk, the conversation that pulls three people out of what they were concentrating on. Neither version is anybody's character. But one of them is invisible and one of them is shared out across everyone sitting nearby, and only one of those shows up as a problem.

So it isn't a location problem. It's a performance problem, and it was one before anybody worked from home. Location policy is just an unusually blunt instrument for addressing it — you constrain everybody in order to manage a few, and the few adapt within a fortnight anyway.

The mixed meeting is where the friction actually shows up, and almost nobody counts it.

Half the people in a room, half on a screen. The room develops its own conversation — the aside, the quick sketch, the exchange of looks that settles something without anyone saying it formally. The person on the wall gets the audio and none of the rest. They are physically present as a rectangle and functionally absent as a participant, and after the third time they stop trying to interject, because the timing is impossible and interrupting a room from a screen requires more social nerve than the point is usually worth.

Then someone summarizes the meeting afterward and their perspective isn't in it, and everyone genuinely believes they were included.

Many teams are distributed now. That's fine. Distributed is workable. Half-distributed, meeting by meeting, without anyone deciding it, is where people get quietly dropped.

And the drifting attention is telling you something, though usually not what people assume.

If somebody consistently cannot hold attention in a particular meeting, that's information rather than a character assessment. Occasionally they're tired. Reliably, in the same recurring slot, week after week? That's a signal about the meeting.

Half a room quietly multitasking is the cheapest attendance record you'll ever get on whether a forum is still earning its place. Everyone present is voting with their attention, and the vote is unanimous and unspoken.

The place is negotiable. The presence isn't — and only one of those two gets written into a policy.

What moves this is making both modes genuinely work rather than picking a winner, and then extending the trust to everybody rather than selectively. Selective trust is worse than none, because it's visible: people know exactly who is being watched, and everyone else adjusts to avoid becoming that person. The rules that matter turn out to be about meetings rather than about buildings.

Cultivating this: Growing the Team, in Part III.

It Breaks in the Joints

Information, decisions and execution each work. Where they meet is where things go missing.

REDUCES FRICTION · WORKS ON DECISION LATENCY

Somewhere in your organization there is a report that is completely accurate and completely useless.

It was researched properly. The numbers are right. It went to the correct distribution list. And it arrived after the decision it was meant to inform had already been made, or worse, after the moment when making it would have mattered had passed unnoticed.

Nobody did anything wrong, and that is the entire problem. The person who wrote it did good work. The person who received it read it. The information moved. It simply didn't reach a decision while a decision was still available.

Information, decisions and execution are usually treated as three separate concerns, owned by three different parts of an organization, improved by three different initiatives. They are not three things. They are one movement with three stages, and it almost never breaks inside a stage. It breaks at the joints.

Information should move, not accumulate.

Most organizations are not short of information. They are drowning in it. Dashboards nobody opens, reports nobody acts on, data lakes filling steadily with things no decision will ever draw from. The scarcity is not information; it is information arriving where a decision is being made, at the moment it is being made.

An organization can double its reporting and become slower, because the cost of information is not the gathering. It is the reading, the interpreting, and the deciding whether it changes anything — and that cost is paid by exactly the people who have the least time.

The useful question is never *do we have this information*. It is *whose decision does this reach, and when*.

Decisions create momentum when they're made.

Every organization has an unspoken clock, and almost nobody measures it: the time between a decision becoming necessary and that decision being made.

It's a strange number, because it belongs to no one. The people waiting are not idle — they've moved to something else, which is precisely why the wait feels free. Nobody bills for waiting. There is no line in any budget for a decision that took eleven weeks, and no report that shows what the organization couldn't do during them.

A useful test: ask how long the last significant cross-team decision took, then ask how long the *deciding* took. In a healthy organization those numbers are close. In most, one is measured in minutes and the other in months, and the gap is not deliberation — it's scheduling, preparation, and waiting for the right people to be in the same place.

The delay is rarely caused by the difficulty of the decision. It's caused by uncertainty about who gets to make it, which is an Ownership problem arriving in disguise.

Execution is the only stage anyone measures.

Which is why it's the stage least often at fault, and the stage that gets the most improvement effort. Delivery metrics, velocity, throughput, utilization — all of it pointed at the one part of the loop that was usually working.

An organization can get significantly better at execution and not move any faster overall, because the constraint was never there. It was three weeks upstream, in a decision that hadn't been made, waiting on a report that arrived correct and late.

The loop closes or it isn't a loop.

Execution produces an outcome. Someone learns something from it — what worked, what didn't, what the estimate missed. If that never travels back to inform the next decision, the organization has a line, not a loop. It will make the same decision again with the same information it had the first time.

That last stage is the shortest, the cheapest, and the one skipped almost universally. It's the subject of the next chapter, and the next chapter is deliberately brief.

None of the three stages is where flow dies. It dies in the space between them: information that never reaches a decision, decisions that never reach execution, execution whose outcome never reaches anyone. Those spaces belong to no one, appear on no org chart, and are therefore nobody's job to fix.

This is the sap. Everything else in Part II describes the tree; this chapter describes the thing actually moving through it — and a tree that stops circulating dies standing up, long before it looks dead from outside.

It almost never breaks inside a stage. It breaks at the joints, which is precisely where nobody owns it.

What moves this is not improving any one stage, which is what each function does and why the joints stay untouched. It is making the joints somebody's — routing information to the decision it serves rather than to a distribution list, moving decisions to where the knowledge already sits, and send-

ing outcomes back to whoever chose. Each of those is a small change at a boundary rather than a programme inside a department.

Cultivating this: Closing the Loop, in Part III.

We Have Solved This Before

The outcome never reaches the people who chose the work, so they choose the same way again.

BOTH · WORKS ON LEARNING CYCLE TIME

Every organization already knows this stage exists. Most have a name for it — retrospective, post-mortem, lessons learned, evaluation — and a document to prove it happened.

The document is usually the problem. Learning that produces an artefact has an obvious place to stop.

Learning is the leaf falling back into the soil. Execution produced an outcome; something about that outcome should change what happens next. Not be recorded. Change something.

The test is unglamorous and immediate: after the last significant thing that went wrong, what is now different? Not *what did we write down*. What decision would go differently tomorrow?

If nothing, the loop is open. And an open loop is expensive in a way that never appears as a cost, because the organization simply pays for the same mistake again at full price, some distance away, where nobody connects the two.

Why the loop stays open

It is almost never because people are uninterested in learning. Three structural reasons account for most of it, and all three are fixable without a programme.

The outcome arrives somewhere other than the decision. The team that chose the work has moved on by the time anyone finds out whether it worked, and the finding-out happens in a different part of the organization — support, sales, an account manager on a call. The information exists. It just never travels back along the path the decision came down. Nobody withheld it; there was simply no route.

The learning is owned by the document rather than by a person. A retrospective produces actions, the actions go into a list, and the list has no owner because the meeting was the owner and the meeting is over. Learning that lives in an artefact has an obvious place to stop, which is why the artefact so often becomes the point.

Admitting the lesson costs someone something. This is the one people are least willing to name. If the honest conclusion is that a decision made by a named person was wrong, and the organization has no way of saying that which does not damage them, the conclusion will be softened until it teaches nothing. What gets recorded is *communication could have been better* — a sentence that has never changed a single decision anywhere.

The third is where psychological safety stops being a pleasant idea and becomes an operating requirement. Not comfort: the specific ability to say what happened without paying for having said it. An organization that cannot do that will produce retrospectives forever and learn from none of them, and it will do so while genuinely believing it has a learning culture, because the meetings are in the calendar.

This is why learning belongs in a chapter about movement rather than a chapter about culture. It isn't reflection. It's the return leg of the circulation — and a tree that draws water up and returns nothing to the soil is not learning slowly. It's dying at a rate nobody has measured yet.

The leaf falls back into the soil, or nothing compounds.

That's the chapter.

What it takes to close the loop is a matter of practice rather than structure, and Part III takes it up there.

Which accounts for every part of the movement: information in, decisions made, execution done, learning returned.

The leaf falls back into the soil, or nothing compounds.

What moves this is smaller than the problem suggests. Route the outcome back to whoever chose the work, by name rather than to a list. Give each lesson one owner and one date instead of eleven actions in a spreadsheet. And accept that the third reason — that the honest conclusion sometimes costs somebody something — is settled almost entirely by what happens the first two or three times a senior person says it about themselves.

Cultivating this: Closing the Loop, in Part III.

CHAPTER 22 · COMPETENCE

The Half-Life of Knowledge

An organization cannot learn faster than the people inside it are learning.

BOTH · WORKS ON LEARNING CYCLE TIME

Somewhere in my thirties I noticed that a particular kind of conversation had started happening to me rather than by me.

A younger colleague would describe an approach to something, and I would recognize the shape of it well enough to follow — and then realize, part way through, that my understanding of the underlying thing was about four years out of date. Not wrong exactly. Just built on a version of the world that had quietly been replaced while I was busy being competent at the previous one.

The first time it was uncomfortable. By the fifth or sixth it had become interesting, which is much the more useful of the two reactions.

Because none of that was anybody's failure. It is simply what happens to knowledge over time. It has a half-life. Not a cliff, not an expiry date printed somewhere — a slow, steady loss of relevance as technology, markets, regulation, customer expectations and ways of working move underneath it. Some of what I learned in 2003 is as true now as it was then. Rather more of it describes a world that no longer exists in the form I learned it.

Artificial intelligence has made that decay noticeably faster, but it did not create it, and I think it is worth resisting the temptation to treat this as a new problem with a new urgency. Anyone who has been in a profession for twenty years already knows this in their body. The work changes continu-

ously, and keeping your competence current became part of the job somewhere along the way without anyone announcing it.

And this is not only a technical matter, which is the part most easily lost.

The technical half is simply the most visible, because a new framework or a new tool announces itself and can be listed on a CV. The rest of it moves just as much and much more quietly. How customers behave and what they now expect as a baseline. What leadership means when a team is distributed across three countries. How you communicate something complicated to people who have every right not to have followed the last four months of context. What a business model can be when the marginal cost of distribution is nothing. How problems get solved now.

I have met people whose technical knowledge was immaculate and whose picture of what customers wanted was assembled in 2015 and never revisited. That is the same decay, in a place nobody thinks to look.

Where this meets Organizational Flow

This paper spends most of its length on structural friction — ownership nobody can name, boundaries that produce handovers, decisions travelling further than the knowledge behind them. All of that is real, and none of it is the whole picture.

Friction also grows when the organization's collective competence stops matching the environment it is operating in. Nothing in the structure has changed. The org chart is the same, the boundaries are where they were, the governance is no heavier than it was last year. And things have become harder anyway.

It shows up in a set of symptoms that look organizational and are at least partly something else:

Decisions take longer, because confidence has quietly fallen and people are less willing to commit to something they only half understand. Specialists become bottlenecks, not because anyone hoarded anything, but because the number of people who can genuinely engage with the work has shrunk. Coordination increases, because uncertainty increases, and uncertainty is expensive to hold alone. More reviews and more approvals start to feel necessary, and they *are* necessary, given what the room currently knows. Change gets harder, in an organization whose structure has not changed at all.

I have watched teams reorganize in response to that, sincerely and thoughtfully, and get almost nothing from it — because the boundary was never the binding constraint. What had actually happened was that the ground had moved and the collective understanding had not moved with it.

That is not a comfortable diagnosis, and I would not lead with it in a room. But it is worth holding as a possibility whenever a structural fix produces less than it should have.

Two kinds of learning, and both are required

The Learning chapter in this paper is about how an organization turns experience into shared knowledge — how what one person found out becomes something the next decision benefits from. That is one half.

The other half sits underneath it and is easy to assume rather than tend.

Continuous competence is individuals acquiring new knowledge from outside the organization. Reading, courses, conferences, communities, side projects, conversations with people

who do not work where you work. It brings in what the organization does not yet contain.

Organizational learning is what turns one person's knowledge into competence more than one person holds. Outcomes travelling back to whoever chose. Lessons that have an owner. Understanding that survives the person who acquired it.

That distinction matters more than it looks, in this paper's terms. Competence belongs to people; a capability belongs to the organization and is only ever as real as the competence currently standing behind it. Learning is how competence gets renewed and spread, and it is the only reason a capability is more than the few people who happen to staff it this year.

Neither is sufficient alone, and the failure modes are different enough to be worth naming.

An organization where people keep learning but the knowledge stays individual does not improve. It accumulates well-informed people who are each slightly ahead of the organization and none of whom can move it. That is a frustrating place to work, and it is where the loud phase of *It Shows Up in People First* often comes from.

An organization with excellent sharing mechanisms and people who have stopped developing gets very good at circulating a fixed stock of knowledge. Everything works. The retrospectives are genuine. And the collective understanding slowly drifts further from the world it is describing, without anything visibly going wrong.

Together they close a loop that renews itself — new understanding coming in, shared understanding forming, better decisions, which produce better experience, which is worth sharing. It compounds in exactly the way friction compounds, only in the direction you want.

The part I find hopeful

I have come to think that having to keep learning is one of the better features of this kind of work rather than a tax on it.

There is a version of a career where you become competent at something and then perform that competence for thirty years. I have met people who wanted that and a few who got it, and it did not look like the good outcome from outside. The alternative — that the ground keeps moving and you have to keep up — is more demanding and considerably more alive. It means the work can still surprise you at fifty-five.

It also puts everybody in the same position, which is quietly levelling. The person with twenty-five years and the person with two are both partly out of date, in different places, and both of them know things the other needs. That is a much better basis for a team than seniority as a proxy for correctness, and it is one of the few genuinely nice consequences of everything changing quickly.

None of which needs a programme. It needs the organization to treat time spent learning as work rather than as a personal hobby subsidized by evenings — and it needs individuals to accept that nobody else is going to own this on their behalf.

The knowledge you have is depreciating, gently, whatever you do. The interesting question is not how to stop that, because you cannot. It is what you are curious about at the moment.

Cultivating this: Growing the Team, in Part III.

EDITOR'S NOTES

The idea that professional knowledge decays rather than accumulating indefinitely has been studied for decades under the heading of skill obsolescence, and the OECD's Skills Outlook and the World Economic Forum's Future of Jobs reports both put numbers to how quickly. I have deliberately not quoted their figures, because the specific percentages move between editions and the argument does not depend on them — anyone who wants the evidence should go to the sources rather than to me. Peter Senge's *The Fifth Discipline* is where the relationship between personal mastery and organizational learning is set out most carefully; this chapter is essentially restating his first discipline in the vocabulary of this paper, and anyone who finds the idea useful should read the original.

CHAPTER 23 · RECOGNITION

No Incident, No Story

Rescue is legible. Prevention is not. Nobody decided that, and it decides a great deal.

BOTH · WORKS ON LEARNING CYCLE TIME

Somebody worked the weekend and saved the release.

It was genuinely bad. The deployment went wrong on Friday evening, three people cancelled what they were doing, and by Sunday afternoon it was working. On Monday there were thanks in the team meeting, and the kind of thanks that everyone can see are meant. At the next review their name came up, attached to the word *stepped up*, and that mattered when the promotion round came around.

All of which was deserved. That is the important part, and it is the part that makes this pattern so hard to shift: nothing in that paragraph is wrong. They did save something. Thanking them was correct.

Now the other person.

In March, somebody looked at the same deployment path, noticed the thing that would eventually fail, and spent a slow afternoon removing it. No incident happened. No weekend was ruined. Nobody sent a thank-you, because there was nothing to thank them for — from the outside, that afternoon looks exactly like an afternoon in which nothing much occurred.

They are not in anyone's review. They cannot be. There is no artifact.

This is a property of how attention works, not a flaw in anyone's character.

Rescue produces a story. It has a beginning, a crisis, a named protagonist and a resolution, and the whole thing fits in ninety seconds at a Monday meeting. It is memorable in the specific way human beings find things memorable.

Prevention produces an absence. An absence has no shape and no narrator, and there is nothing to be memorable about. The person who prevented the incident cannot even describe what they did without sounding like they are claiming credit for a hypothetical.

Nobody decided to reward it this way. No leadership team has ever agreed to value rescue over prevention. It is simply what happens when recognition follows attention and attention follows events.

What an organization is actually selecting for, over years, is people who are good at rescue.

That takes a while to show up and is hard to see when it does.

The people who are good at rescue get noticed, promoted, and given more of the work that requires rescuing. The people who are good at prevention do it anyway, out of temperament, until they get tired of it or leave — and when they leave, nobody connects their departure to anything, because they had no incidents attached to their name either.

Ten years of this produces an organization with a great deal of quiet institutional knowledge about how to survive incidents, and remarkably little about how not to have them. It has a well-drilled crisis routine, people who are calm under pressure, and a genuine competence at recovery that everybody is rightly proud of.

It also has, somewhere in it, the same three failure modes it has been recovering from since 2019.

And it teaches the next generation faster than anything written down.

A new engineer works out the incentive in about a quarter. Not from anything anyone says — from watching who gets thanked. The lesson is not cynical and does not feel like a lesson: it is simply an accurate reading of what this place notices, and people act on accurate readings.

Which is why exhortation does not touch this. You can say prevention matters, in writing, at an all-hands, sincerely, and it will lose to what happened in the meeting on Monday.

It shares a mechanism with a pattern elsewhere in this part, and is not the same one.

Maja is a *routing* dependency: a capability reachable only through one person, costing her interruptions and the organization a risk it has decided to admire. This is a *reward* dependency: the organization pays for rescue, so rescue is what it gets more of.

What they have in common is worth saying once. In both cases the person absorbing the friction is the person being rewarded for absorbing it — which is precisely why the friction never gets fixed. Everybody in the arrangement is doing well out of it, right up until the moment they aren't.

The most valuable work in most organizations produces no evidence that it happened.

What moves this is not thanking prevention more loudly, which reads as consolation and fools nobody. It is making the absence visible — asking what did not happen this quarter, and who made it not happen — and then treating that answer with the same seriousness as the incident report. Somebody senior has to go looking, because by construction nobody will come and tell you.

Cultivating this: Leading Without Deciding, in Part III.

CHAPTER 24 · VALUE

Everything Was Green

Every report accurate, every project delivered, and nothing the organization can do is any better.

CREATES VALUE · WORKS ON LEARNING CYCLE TIME

I have sat through a great many progress meetings in large organizations, and they have a texture you can recognize anywhere. A long call or a long room. Fifteen or twenty teams, projects and products, reporting in sequence, three or four minutes each. A wall of status behind them.

Almost everything is green.

Amber turns up occasionally, and amber has its own etiquette — it arrives with a recovery plan attached and a quiet assurance that it will be green again by the next one. Red is rare enough to count as an event, and usually means something has already gone wrong publicly. A task moves to the next sprint, which is described as a re-plan rather than a delay. Nobody finds any of this strange, and they are right not to. It happens every fortnight.

Then listen to what is actually being said in those four minutes. It is nearly all production. What was finished, what is in progress, what moved, what is coming. Occasionally a genuine blocker, phrased carefully enough that it does not read as an accusation about another team.

What almost never comes up is whether any of it did anything for anyone.

If you sit in one of these this month, keep a rough tally: minutes spent on what has been produced, against minutes spent on what changed for somebody outside the building. It is

rarely a close contest, and the ratio may be the most honest number in the meeting.

Nobody is hiding anything. Every person in that room is reporting accurately on what they were asked to report on, in a format built to answer a question about progress. It answers that question well. It was simply never built to ask the other one, and after a few years of everyone answering it correctly, the organization has an enormous amount of evidence that things are going fine.

That is the pattern this chapter is about, and it is the most common one in the paper: every project green, every roadmap item shipped, every quarterly review satisfied — and what the organization could actually do for its customers no better in December than it was in January. The question of what any of it produced was answered once, at funding, in the one moment when no evidence existed yet.

The word underneath all this deserves a definition, because it has been doing quiet work for twenty-odd chapters without being examined, and unexamined is exactly how it usually travels. Value is the most-used and least-defined word in organizational language — which is why it can appear in a strategy document without anyone disagreeing, and why almost nobody in the building could tell you what it meant if you asked them separately.

So, plainly: value is what someone on the other end actually got. Not what was delivered. What landed.

Two words are worth separating, and the paper uses them consistently from here on.

Output is what was delivered. The release that went out, the report that was written, the migration that completed. Countable on a Thursday by somebody in the building, and entirely a fact about us.

Outcome is what changed for the person on the other end. Whether they ended up better off. That is a fact about them — and it is the same thing this paper means by **value**, used interchangeably because they are the same thing seen from two angles: outcome is the word that fits in a funding conversation, value is the word that fits in the argument about why any of it matters.

The distinction sounds like hair-splitting and is the entire chapter. Output is a fact about you. Outcome is a fact about them.

Which has a consequence worth stating plainly, because it decides who can be held to what. A team can be held to its output, always. It can be held to an outcome only if it is close enough to the person on the other end to find out — and if it is not, then asking it to own the outcome is asking it to own a rumour.

Which is why organizations measure delivery instead. Delivery is legible from the inside — it has dates, owners, a definition of done, and a satisfying moment where something moves to the right on a board. Value is legible only from the outside, arrives late, and refuses to attach itself cleanly to any one team's effort. Given a choice between a number you can produce this week and a number you can't, the week wins, every time, in every organization I have worked in.

In the tree, value is the canopy. It's worth being precise about why that image is the right one rather than a decorative one.

The canopy is the only part of the tree the outside world experiences. It's also the part you can do least about directly. You cannot instruct a canopy. You can improve soil, protect roots, keep the sap moving, and then the canopy is the honest result of all of it — an *output* of the system in the strict sense, not a target the system was aimed at. Every attempt to intervene on the canopy directly is either cosmetic or damaging.

Organizations do it anyway. Pushing on value directly looks like pressure on delivery dates, on volume, on visible output — all of which produce more canopy in the short term and thinner roots underneath, which is a trade you only notice one or two seasons later, by which time the people who made it have usually moved on.

And this is where *sustainable* earns its place in the phrase, rather than sitting there as a comfortable adjective.

Value can be extracted or it can be grown. Extracted value is real — it shows up in the quarter, it is genuinely there — and it is taken from a capability that was not replenished. Grown value arrives because the conditions underneath it kept working, which means it arrives again next season without anyone staging a heroic effort. Both look identical on the way up. They only diverge later, and the difference is entirely in whether the loop closed: whether the outcome travelled back and changed what happens next, or whether the organization simply spent what it had.

An organization with flow produces value as a by-product. An organization without it produces value in bursts, each one paid for by somebody, and each one followed by a recovery period nobody plans for.

Which brings this back to friction one last time, because the connection is more direct than it first appears. Every shape friction takes in this paper — ownership nobody can name, context rebuilt at each boundary, decisions waiting on a forum, outcomes that never travel back — is a place where effort was spent and no value arrived at the far end. That is not a metaphor. It is the actual arithmetic. The work happened. The salary was paid. Nothing reached anyone outside the organization as a result.

Friction is the difference between what an organization spends and what it delivers. Flow is what closes that gap. Value is what comes out of the other side, and the only reason any of the rest of it is worth doing.

Every observation in this part has been one argument approached from a different angle, and every one of them came down to clarity or cultivation — named back in The Cost of Value Never Created, before either of those words existed yet for you.

What moves this is contact rather than measurement, though measurement follows. Somebody in the organization has to be close enough to a person on the receiving end that the numbers can be contradicted by a specific human being — and that channel is cheap, informal, and the first thing cut when the quarter gets tight.

Cultivating this: Creating Value, in Part III.

CHAPTER 25 · SYSTEMS

It Grew Like That

Nobody designed the shape the organization has now. It arrived one reasonable decision at a time.

BOTH · WORKS ON ALL FOUR

Ask why a particular team sits where it does, in any organization that has been going a while, and you will get a story rather than a reason.

It used to be part of a different division. It came in with an acquisition and never quite got absorbed. It reported to someone who has since left, and moving it now would mean reopening a conversation that took four months last time. My favourite version of this answer, given entirely without irony: *it has always been there.*

Nobody sat down and designed that shape. It arrived one sensible decision at a time, each of them defensible on the day it was made, and the sum of them is a structure that no single person would have drawn on purpose and that everybody now works inside.

I find that oddly reassuring, and I think it is the most useful thing to understand about organizations. If the shape had been designed, someone would have to be blamed for it. It wasn't. It grew.

Which is worth sitting with, because it changes what kind of thing you are looking at.

A structure that was designed can be redesigned. You find the person who drew it, establish what they were solving for, and decide whether it still holds. That is a tractable afternoon.

A structure that *grew* has no such person. There is no original intent to consult, because there was never one intent — there were forty, spread over a decade, each of them local and reasonable and mostly forgotten by whoever had it. Asking "why is it like this" is the wrong question, and it is the one everybody asks first. The answer is always the same and always unsatisfying: because of a sequence of things, most of which are no longer true.

Which is why organizations reorganize so confidently and change so little. Redrawing the chart is treating the shape as a design. The shape is a residue.

None of that makes it permanent. Grown things can be tended, cut back, and grown differently — that is most of what the rest of this paper is about. But it starts from noticing that nobody is defending the current arrangement, because nobody chose it. There is far less resistance waiting than most people expect. There is just nobody whose job it is to notice.

Some of the most expensive structures in an organization are the ones nobody ever decided on.

Cultivating this: Tending What Grew, in Part III.

EDITOR'S NOTES

That organizational arrangements work in relation to their context rather than in themselves has been arrived at from several directions — sociotechnical systems research, contingency theory, and systems thinking all converge on it. This chapter is restating a well-supported finding in its own vocabulary rather than discovering one. Where I would resist going further than the evidence allows is on causation. It would be tidier to say the conditions are the cause and the structure merely the result, and that is too clean: a new structure changes incentives, communication paths, and what people do by default, which over time changes the conditions in turn. The two shape each other. That is why the text says the shape was *part* of the result rather than the whole of it.

The Conditions Don't Come in the Box

The shape was part of the result. What produced it stayed behind.

BOTH · WORKS ON ALL FOUR

Somebody comes back from a conference, or finishes a good book, and what they bring back is a diagram.

It is usually a genuinely good diagram. That is what makes this one durable — nobody returns with a bad idea. They return with something that demonstrably worked somewhere, drawn clearly enough to put on a slide, and the meeting that follows is one of the more enjoyable meetings of the quarter. Everyone can see it. Everyone can see how it would fit here. Somebody volunteers to lead it.

Eighteen months later the boards are still on the wall and nothing moves any faster than it did.

I have been the person with the diagram. More than once, and with real enthusiasm, which is why I do not think this is about people being credulous. The diagram was accurate. It described something that was working. What it could not describe was everything holding it up.

The visible shape of an organization is a result, not a recipe.

The two most-copied models of the last few decades both make the point, and both make it in the same way.

Toyota's production system does not arrive in the boards and the daily stand-ups. Those are what it looks like from outside — the parts that photograph well. What makes it work is the capability underneath: decades of building people who can see a problem, stop the line, and be thanked rather than

blamed for it. Lean programmes have been installing the artifacts without that capability for thirty years, and failing on exactly that distinction, usually while reporting successful adoption.

Spotify's engineering model was published as a snapshot — how one company happened to be working at one moment, written up by people who have said since, repeatedly and in public, that it was never meant to be adopted as a framework. It was adopted as a framework anyway, in a great many organizations whose conditions had nothing in common with Spotify's, several of which then concluded that squads and tribes do not work.

Squads and tribes worked fine. They were the shape of something, and the something did not come with them.

What actually gets copied is the part that is easy to see.

This is not a failure of intelligence, it is a property of distance. From outside an organization you can observe its structures, its ceremonies, its vocabulary and its org chart. You cannot observe how long a decision takes, who is allowed to say no, what happens to the person who raises a problem in week two, or which of the written rules everyone quietly ignores.

So the transferable-looking parts are precisely the parts that were consequences, and the parts that did the work are invisible from where the observer is standing. You end up importing the canopy and wondering why it will not stay green.

And the borrowed shape usually lands on soil that already had its own.

The conditions in the receiving organization are not neutral ground. There is already a way decisions get made, already a set of things it is unwise to say in a large meeting, already a funding rhythm that predates anybody in the room. The new

model arrives on top of all that, and what emerges is a hybrid nobody designed: the ceremonies of one organization running on the incentives of another.

That hybrid is frequently worse than either. It has the overhead of the new thing and the constraints of the old one, and because everybody agreed to adopt it, questioning whether it fits reads as questioning the decision.

The honest version of this is uncomfortable for anyone who writes things down.

It applies to this paper too, which is worth saying plainly rather than leaving as an unstated risk. The examples here came from particular organizations with particular conditions, and they were never meant to be lifted. Anything in these pages that reads like a shape you could install is either badly written or being read as something it is not.

What travels between organizations is not structure. It is the question somebody asked that made the structure obvious — and questions are cheap to carry, survive translation, and cost nothing if they turn out not to apply.

Borrowing the answer is what disappoints. Borrowing the question rarely costs anything.

Cultivating this: Tending What Grew, in Part III.

CHAPTER 27 · LEADERSHIP

Decided Upstairs, Felt Downstairs

Friction gets designed in rooms where nobody believes they are designing anything.

BOTH · WORKS ON OWNERSHIP CLARITY

Ask a management team whether they make architectural decisions and they'll say no. Architecture is technical. It happens somewhere further down, among people with a different vocabulary.

Then they spend the morning deciding which departments exist, who reports to whom, and where one team's responsibility ends and another's begins.

That *is* the architecture. Not a factor shaping it — the thing itself. Split one capability across two departments and you have guaranteed a handover, a negotiation and a delay, permanently, in everything that capability touches. The decision took four minutes. It will outlive everyone who made it.

Reorganizations are architecture, performed by people who were told architecture was somebody else's specialism.

Which means the most consequential design work in most organizations is done by the group least likely to review it as design work. Nobody sketches the alternative. Nobody asks what the new shape makes slow. The boxes move, the announcement goes out, and the consequences show up eighteen months later as a culture problem.

Once you see it in the org chart, you start noticing it in the other commitments leadership makes, and budgeting is the clearest second case.

Look at what a budget is actually attached to. Almost always a project or a system — something with a name, a scope and an end date. Which is entirely reasonable: those are the things that can be scoped, costed and closed, and finance quite properly needs something it can close.

But a project ends and a system gets replaced. The capability underneath them doesn't. Claims handling outlives every claims system; onboarding outlives every onboarding project. So the money attaches to the temporary thing, and the enduring thing — the one the organization actually needs to keep getting better at — has no line of its own and no one accountable for it across the years.

That has a second effect worth noticing. When funding is granted to a project, the project becomes the thing that gets reported on: delivered, on time, within budget. All three can be true while the capability is no better than it was. Nobody misrepresented anything. The question of what the investment was meant to *produce* simply wasn't the question the process was built to ask, because it was answered once at allocation — the one moment when no information exists yet.

Add the annual rhythm and you get the mismatch from The Cost of Value Never Created in its purest form: the work needs a decision this week, and the funding mechanism can produce one in April.

None of this is anyone's failure. It's a set of arrangements that made sense when what you built stayed built — and that quietly stopped matching how the work actually behaves.

And then there is the assumption underneath both: that a manager is where decisions go. Not someone who creates the conditions for decisions to be made well, but the person who personally makes them. It's rarely stated, it's frequently rewarded — we still measure a leader's standing partly by how many people report to them — and it produces a specific, predictable trap.

If every decision must travel upward, everyone above needs enough information to decide. Getting the right information to the right level takes people whose job is moving and summarizing it, so you add a layer. But each layer sits further from the detail than the one below, so the summaries thin out while the decisions stay just as consequential. Add enough layers and you arrive somewhere nobody intended: the people with the authority to decide have the least direct knowledge of what they're deciding, and the people with the knowledge have none of the authority.

Nobody designs that. It's simply what you get by default when decisions travel upward and information has to chase them.

That trap has always had a quiet exception built into it: the summarizing itself needed a person, because nobody else had the time to do it. That exception isn't permanent. A system can already read what ten teams produced last week and hand a decision-maker the three things that actually matter, without two days spent assembling the deck first.

I want to be careful here, because this isn't an argument that the layer was pointless. Reading the shape of a room, translating between how a team talks and how leadership needs to hear it, protecting people from noise they don't need — most of what a good manager in that position does was never really about moving information. It was judgement, exercised while moving it. That part doesn't go anywhere.

What does thin out is the part of the role that was purely relay — summarizing because summarizing was the only way the information could travel at all. Almost nobody was hired for the relaying. Most people in that seat were put there for their judgement, and the relaying was simply what the structure demanded of them alongside it, quietly, until it looked like most of the job.

So if this describes a seat you're in, the question worth asking isn't whether you're replaceable. It's whether the organization around you has ever actually seen the judgement you carry, separate from the reports that judgement got wrapped in to travel upward. Most organizations haven't, because the reports were the only part anyone could see. That's worth raising on its own, regardless of what technology does next — not because your position is at risk, but because the best part of what you do has been invisible for reasons that had nothing to do with its value.

Every one of these is a design decision that was never recognized as one, which is the whole argument of this chapter.

What moves this is treating the rooms as what they are. Before a reorganization is announced, name the three things it will make slower — there always are three, and if nobody can name them the design has not been examined. Fund capabilities for periods rather than projects for dates. Push one decision down and do not take it back the first time it goes badly. None of it needs a mandate, and all of it requires giving something up, which is why it is harder than it looks. What to do about it is a matter of practice, and Part III takes it up.

Cultivating this: Leading Without Deciding, in Part III.

The Long Way to a Yes

The decision is small. The distance to the judgement it needs is not.

REDUCES FRICTION · WORKS ON DECISION LATENCY

A note on the word, for anyone dropping in here. This paper uses ***architecture*** *in a wider sense than the technical one — the deliberate shaping of how an organization is put together, of which system design is one part. A reorganization is an architectural decision. So is a funding model, a team boundary and a reporting line. What a Capability Is, and Is Not in Part I sets out the vocabulary this rests on.*

I helped build the reputation architecture still carries in some organizations. Ownership has the story: a board created to review significant decisions, which people stopped bringing decisions to, which slowed everything down while seeing less than before.

The lesson I drew was not the one I went looking for. The problem was not that we reviewed too much or too little. The problem was that architecture had been made into a *place*.

Architecture is a competence, not a role.

And it is worth being exact about what the competence is for, because the field has spent thirty years defining itself by its instruments rather than its purpose.

Architecture is the practice of reducing structural friction.

It is not primarily the design of technology. Nor is it the design of organizations. It is the continuous cultivation of the conditions under which value gets created.

Diagrams, target states, standards, reference models, governance forums — all of those are instruments architecture may reach for, and each is useful in the narrow case where it earns its cost. None of them is what architecture *is*. A beautiful landscape model of an organization that still cannot decide anything is the instrument delivered and the job skipped.

The moment it becomes a role, three things follow automatically. Decisions have to travel to reach it, which adds delay. The people who make decisions daily stop developing the judgement to make them well, because someone else is responsible for that. And whoever holds the role, now at a distance, starts optimizing for consistency — the only thing visible from a distance — rather than for whether the organization can change.

That last one is the quiet damage. Consistency is not a bad goal, but it is the goal you drift toward when you can see the shape of things and not the cost of changing them.

A note on who *whoever* means, because the word architect does more harm than good here. In some organizations this work sits with people carrying that title. In others it is an engineering manager, a head of product, a CTO, or a founder who has never once thought of themselves as doing architecture. In smaller companies it is often whoever happens to have the longest memory of why things are the way they are. The title varies; the work does not. Somebody in every organization is already deciding which boundaries hold, which decisions get escalated, and what becomes expensive to reverse — deliberately or, more often, as a side effect of deciding something else.

Which is the argument for spreading the competence rather than concentrating it. If this were genuinely a specialist function, keeping it in one department would be defensible. It isn't. The decisions that most shape whether an organization can change are made in budget rounds, in reorganizations, in board meetings about acquisitions and operating models —

rooms where nobody present would describe the conversation as architectural, and where the cost of getting it wrong takes two years to show up and is attributed to something else by then. That competence belongs in those rooms, held by whoever is already sitting in them.

Here is the standard I'd use instead, and it's the only one I've found that survives contact with a real organization:

Architecture should make the organization, and what it builds, easier to change.

Worth being precise about what kind of claim that is, because it sits at a different level than "reducing structural friction" above, not in competition with it. Reducing friction is the purpose — the reason the competence exists at all, so more of what the organization spends actually reaches value. Easier to change is the test applied to one decision at a time, because it is the sharpest question I have found for telling in advance whether a specific choice is moving toward that purpose or quietly working against it.

None of this makes security, compliance or consistency optional, and I don't want to be read as saying it does — a design that fails them isn't a design, whatever else it gets right. What changes is where they sit. They are constraints any real solution has to survive, settled the way the guardrails chapter in Part III describes. They are not what the practice organizes around. An architecture optimized for consistency or compliance and nothing else will produce both, reliably, right up until the organization needs to become something those constraints didn't anticipate — which is the one thing every organization eventually needs to do.

That standard is unusually testable. Take any architectural decision, in the broad sense this paper uses — a system choice, a team boundary, a reporting line, a funding structure — and ask what it makes harder to reverse. Then ask whether the thing being made irreversible is one you're confident enough about to be stuck with.

Most bad architecture isn't wrong so much as premature — a reasonable decision made irreversible before anyone knew enough to make it permanent.

And this is where the competence has to live in the teams rather than above them, because the person closest to the work is usually the only one who can see which decisions are cheap to change and which quietly aren't. That judgement doesn't transfer through a review meeting. It has to be grown where the decisions are made.

Which brings the chapter to the claim underneath all of this, and the one Part III is built on: **architecture is not the practice of designing solutions. It is the practice of cultivating the conditions where good solutions keep emerging.** If architecture is cultivation rather than design, it was never going to be a place solutions get sent to.

One boundary is worth drawing before this goes any further, because a chapter this expansive about architecture invites the wrong conclusion. **Architecture can influence Organizational Flow. It cannot own it.** Flow is produced by the organization as a whole — through decisions made in leadership, in teams, in how capabilities are funded and who is trusted to decide. Architecture is one competence among several that shape those conditions, and a paper that let it stand in for the whole would be making the same mistake as the board in the story at the top of this chapter.

A decision that has to travel to find its judgement will arrive late, changed, or not at all.

What moves this is shortening the journey by moving the judgement rather than speeding up the queue — writing down the answer once with the reasoning attached, being present in the rooms that do not think of themselves as architectural, and converting recurring reviews into things a team can carry. Governance asks *was this decision approved*.

Architecture, done as a competence, asks *are the people making these decisions equipped to make them well* — and the slow work of making that true is what Part III is for.

Cultivating this: Equipping, Not Reviewing, in Part III.

EDITOR'S NOTES

The standard this chapter settles on — that architecture should make things easier to change — has a much more rigorous statement elsewhere. *Building Evolutionary Architecture* (Neal Ford, Rebecca Parsons and Patrick Kua) makes guided, incremental change the first property an architecture should have, and then does the part I do not: it operationalizes it, through fitness functions that make the properties you care about continuously and automatically verifiable. What I am doing here is applying the same standard one level out, to organizational decisions — team boundaries, reporting lines, funding structures — where there is nothing to automate and no equivalent of a fitness function to lean on. That makes this chapter weaker as engineering and, I think, useful anyway, because the decisions that most often make an organization hard to change are not the ones in the codebase. Anyone applying the idea to systems should go to them for the mechanics.

CHAPTER 29 · INTERFACES

You Have to Know a Guy

A capability you can only reach if you happen to know the right person.

BOTH · WORKS ON HANDOFF LOSS

A new starter, second week, needs something from another part of the organization. They do the sensible thing and look for how to ask.

There is a page on the intranet, last updated three years ago, linking to a form that submits to a queue nobody monitors any more. There is a shared mailbox. There is a channel with eleven members and no messages since March.

So they ask their own team, and their own team says: oh, just message Maja.

Maja is excellent. Maja knows how everything works, answers quickly, and has done for years. Nobody in this story has done anything wrong, and the newcomer has just learned the actual interface, which is not the form and not the mailbox but a person's name, passed along by word of mouth like a good restaurant.

This is the most cheerful of the patterns in this part, because it looks so much like things working. Somebody helpful, helping. It only shows its price later — when Maja is on holiday, or in a workshop, or has moved to another company and taken the routing table in her head with her. Then a capability that seemed perfectly available turns out to have been available through one person the whole time.

The part I overlooked for years is what it costs Maja.

She is interrupted perhaps fifteen times a day by questions only she can answer quickly. Each one takes four minutes and costs her twenty, because the work she was doing before the interruption has to be picked up again from wherever she dropped it. Her own deliverables slip, and she makes up the time in the evening. Meanwhile the thing she is actually paid to be good at gets whatever attention is left over, which by Thursday is not much.

And she will not complain, because being the person everyone comes to is genuinely lovely. It is a form of standing you cannot get any other way. She is visibly useful, obviously indispensable, thanked constantly — and at the next review somebody will say *we can't afford to lose Maja*, and mean it as praise. Which it is. It is also an accurate description of a risk the organization has decided to admire rather than address.

That is what makes this pattern so durable. Everyone involved is being rewarded: the newcomer gets a fast answer, the organization gets a working shortcut, and Maja gets to be the one who knows. Nobody has any incentive to notice that none of it scales, that all of it lives in one person's head, and that the day she moves on, the organization will discover which of its capabilities were real and which were her.

What happens when nobody finds Maja

The newcomer above got lucky. Somebody told them a name. The more expensive version is when nobody does.

A team needs something, looks for it, cannot find it, and reasonably concludes it does not exist. So they build it. They are competent, they are under time pressure, and they bend a rule or two on the way because the rule was written for a route they never found. It works.

Now there are two endpoints doing more or less the same job. Not identically — one rounds differently, one treats a cancelled order as still open for a day, one was built before a rule changed and nobody told its author because nobody knew it existed. Both are correct, in the sense that each does exactly what its team intended. The organization now has two answers to the same question, and no way of knowing which one anybody is looking at.

The subtler version needs no new system at all. A team is given credentials to something that already exists, reads the data, and works out for themselves what the fields mean — because there was nothing to read and nobody obvious to ask. Their reading is sensible and slightly wrong, in the way any honest reading of an undocumented thing is slightly wrong. They build reporting on it. Six months later there is a third version of the truth, in a deck, in front of people making decisions with it, and it disagrees with the other two by a margin nobody can explain.

What is worth noticing is that no rule was broken in the second case at all. Access was granted properly. The team did what access implies they should do. What was missing was not permission or competence — it was anything that said *here is what this means*, and in its absence people supply their own meaning, which is what people do.

Both of these usually surface years later, as a data quality problem, and get handed to someone to reconcile. It rarely gets recorded as an interface problem, which is what it is: the first thing anybody could not find was the front door.

What Maja actually is, in the vocabulary An Interface Nobody Can Find sets out, is the bottom rung: not automated, not self-service, not even a queue with a name on it — a person you have to know. And the reason it persists is that from the inside it feels like the top rung, because for anyone who knows her the answer arrives immediately.

Both versions are the same mechanism. Once with a helpful person standing in the gap, once without. The only difference is whether the cost lands on Maja or on the second system nobody knew they were building.

What moves this is making the capability reachable by somebody who knows nobody — which sounds like documentation and is really about deciding that *how to use this* is part of the thing rather than a description of it. Maja stops being load-bearing at the point where a stranger can get what she gives them without her, and not one moment before.

Cultivating this: Opening the Front Door, in Part III.

CHAPTER 30 · DISCOVERY COST

The Detective Work at the Front of Everything

The weeks each initiative spends establishing the facts before anybody starts, redone from scratch every time.

BOTH · WORKS ON DECISION LATENCY

Before an initiative can start, somebody has to find out what is true. This is the cost I would price first if I could only price one, and it is the one nobody has ever put in a plan.

Nothing can start until somebody has established the facts. Does this already exist somewhere? Who owns the thing we need? Is that a person or a forum, and when does the forum next meet? Where does the data live and what do the fields actually mean? Who has to say yes, and what will they want to see before they do?

None of that is the work. It is finding out what the work will involve — and it gets done fresh, by different people, for every initiative, because nobody wrote it down last time either. The people doing it are usually rather good at it, which is part of the problem: an organization full of competent detectives feels resourceful rather than slow.

A few weeks is unremarkable in a large organization. It appears in no plan, because it does not look like a phase. It looks like getting started.

The version that gives the game away is the proof of concept. A PoC is meant to be the cheap fast thing you do to find out whether an idea has legs — a fortnight, in principle, and the whole point is that being wrong costs almost nothing. So ask

how long it took from somebody first saying *we should try this* to anybody actually trying it. The gap is routinely longer than the PoC itself, and nearly all of it is detective work.

Which produces an effect nobody intends and nobody decides on: it raises the bar for what is worth attempting at all. If finding out costs three weeks before anything begins, small ideas stop clearing it. They are not rejected — nobody rejects them, there is no meeting — they simply never get proposed, because the person who had the idea did the arithmetic in their head and decided not to bother. What survives is the large initiative, which is slower, more expensive, and considerably harder to be wrong about cheaply.

And the facts do not stay found, which is the part that turns a one-off cost into a recurring one.

The version I have seen most often is small and almost polite. Something moves — an endpoint, a report, a field, a shared folder, a responsibility — and the move is announced to the people who were in the room when it was decided. Everybody else finds out by trying to use the old thing and discovering it is not there.

Nobody hid anything. There was probably even a message, in a channel, in March. But a change communicated once, to whoever was present, is not the same as a change anybody downstream can find, and the second group is always larger than the first.

The other version is quieter still: the thing is exactly where it was, and the way in has gone. A credential expires. A permission is tightened during a security review, correctly. The person who granted access last time has moved on and nobody inherited the ability to grant it. The capability has not changed at all — it has simply become unreachable, and from the outside those two look identical.

Both restart the detective work from the beginning, for everyone except the people who already knew. And both are usually discovered by somebody in the middle of something

else, which is why they get handled as an interruption rather than recorded as a cost.

Which means the discovery tax is not paid once per initiative. It is paid again every time something moves without leaving a forwarding address.

An organization that cannot afford to try small things ends up making only big bets. That is not a strategy anyone chose. It is what a discovery cost does to a portfolio, quietly, from underneath.

Cultivating this: Opening the Front Door, in Part III.

An Interface Nobody Can Find

Two standards for the same need, and a form that made the waiting tidier rather than shorter.

REDUCES FRICTION · WORKS ON HANDOFF LOSS

Every capability is reached somehow.

Someone or something asks it to do its work, and the way that asking happens is a design decision — usually an unexamined one, and one that determines more about how the organization moves than the capability's own quality does.

There are only two fundamental shapes. **Synchronous:** ask, and wait for the answer before continuing. **Asynchronous:** announce what happened, and carry on without waiting.

The difference sounds technical and isn't. Synchronous means one capability's availability has become another's availability — if the thing you're calling is slow, you are slow; if it's down, you're down. You've coupled the two in time, which is the most expensive kind of coupling there is because it can't be scheduled around. Asynchronous decouples them and charges you differently: you give up certainty about *when*, and you take on the work of handling things arriving out of order or twice.

Neither is correct. What's incorrect is choosing without noticing you chose — which is what happens when the shape of the interface is inherited from whatever the first integration happened to do.

But the friction that costs most isn't the shape. It's the hiding.

Consider what it takes to use a capability someone else owns. In a healthy organization: you find it, read how it works, and use it. In most organizations: you ask around until you find out who owns it, book time with them, explain what you're trying to do, discover three special cases that aren't written down anywhere, and build against a shared understanding that exists only in the memory of the two people who had the conversation.

That conversation is a handover. It happens every single time anyone new needs the capability, forever, and it never appears as a cost because it looks like helpfulness.

Then the special cases accumulate, and they accumulate for a specific reason: when nobody can see how something was meant to be used, everybody uses it slightly differently, and every one of those differences has to be supported afterwards. Undocumented interfaces don't stay simple and undocumented. They become complicated *because* they were undocumented.

An interface nobody can find isn't an interface. It's a person you have to know.

Which brings up the distinction most organizations get backwards.

Internal and external interfaces are usually treated as two categories with two standards. External ones get documentation, versioning, examples, a support commitment, and someone who thinks about whether they're pleasant to use. Internal ones get a Slack message and the name of whoever built it.

The distinction is false, and the cost of believing in it is paid daily. An internal consumer has exactly the same needs as an external one — to discover that the thing exists, understand what it does, and use it without a meeting. The only difference is that internal consumers can't take their business elsewhere, so they absorb the friction quietly instead of complaining, and the organization never learns what it's paying.

A form is not self-service. Neither is a shared mailbox, a ticket queue, or a request channel. Every one of them is a handover with a nicer entrance. Somebody still has to read it, work out what was actually meant, decide whose it is, and get round to it — and the person who submitted it is still waiting on a human being they cannot see and cannot chase. What the form did was make the waiting tidier and much easier to report on, which is precisely why queues survive the initiatives meant to remove them.

Self-service is when the person asking gets what they need without anyone else being involved. Something they can call. Access they can grant themselves inside agreed limits. A dataset they can query, with the meaning of the fields written down next to it. The test takes a second: if the request lands in somebody's list, it is a queue.

It is worth naming the whole ladder once, because the rest of the paper keeps landing on one rung or another of it. A thing can be **automated**, so nobody is involved. Or **self-service**, so the asker gets it without entering anyone's list. Or a **queue**, where somebody works through requests in some order. Or **a person you have to know**, which is not really a rung so much as what is left when nobody built the others.

Most organizations have a few things at the top, a great deal in the middle, and considerably more than they realize at the bottom.

None of which is an argument against queues. Plenty of things need judgement, and a well-run queue with a name on it is far better than not knowing who to ask. The cost is in the mislabelling — an organization that believes it has built self-service stops looking, and the waiting carries on underneath, now with a dashboard on top of it.

What moves this is treating an internal consumer exactly as you would treat somebody who could take their business elsewhere — because the only difference between them is re-

course, not need. That is a standard rather than a project, and it is applied one capability at a time.

Cultivating this: Opening the Front Door, in Part III.

Everyone Carries the Strictest Requirement

A rule that belongs to one part of the organization has a way of becoming a rule for all of it.

BOTH · WORKS ON DECISION LATENCY

Somebody once explained to me why a small internal tool, used by eleven people to track a training schedule, had taken nine months.

The explanation was entirely coherent. It had to go through the same review as anything touching customer data, because the review process did not distinguish. It had to run on the approved platform, because there was one approved platform. It needed a data classification, an architecture sign-off and an entry in a register, all of which existed because the organization handles genuinely sensitive information and had been fined once for handling it carelessly.

None of that was stupid. Every one of those controls was put there by somebody thoughtful, in response to something real. And a training schedule for eleven people had cost nine months.

That is the pattern, and once you have seen it you find it everywhere.

Different parts of an organization genuinely need different conditions.

They handle different information and carry different risk. The consequences when something goes wrong are not comparable. Some sit inside heavy regulation and some sit

nowhere near it. Some need extraordinary resilience or security; others would gain far more from being able to try something on a Tuesday.

But constraints are written once and applied broadly, because writing one rule is much easier than writing seven and deciding which applies where. Over time the requirement that belongs to the most sensitive corner of the organization becomes the requirement everybody works under.

I have watched this most clearly in regulated environments. Banks, insurers and public institutions have excellent reasons to care about information, security, continuity, outsourcing, jurisdiction and third-party risk — and not one of those reasons applies equally to every capability they run.

If the organization is treated as one homogeneous thing, the most cautious corner ends up setting the conditions for all of it. Controls designed for the exceptional case become the ordinary case. Decisions move upward. Approvals multiply. The range of technologies anyone can even suggest narrows quietly, until people stop suggesting.

Not one of those decisions looks unreasonable on its own. Together they are structural friction, and the organization experiences it as caution rather than as cost — which is why it is so rarely questioned. Nobody wants to be the person arguing for less care.

It happens in the infrastructure too, and there it is harder to see

The same mechanism runs one layer down, where it is even less visible because it looks like efficiency.

Shared infrastructure is genuinely good. A common platform lets teams inherit security, identity, observability and deployment without rebuilding any of it, and organizations

that have one move visibly faster than those that do not. That is not the problem.

The problem starts when everything runs through the same database, the same integration layer, the same runtime, the same deployment process — because at that point the platform can only be configured for its most demanding tenant. The workload that needs the strictest data residency sets the residency for everyone on it. The system that cannot tolerate downtime sets the change process for everyone on it.

Infrastructure that was shared for efficiency has quietly become shared friction, and the accounting never shows it, because the platform genuinely is cheaper than seven platforms would have been.

The related one is the choice nobody expects to revisit

There is a second version of this that spreads just as far, and it is quieter because it arrives looking like a decision rather than a rule.

Somebody chooses a platform, a provider or a SaaS product. It is the right choice at the time — it meets the need, it is secure, it is faster to start on than anything built in-house, and the alternative would have cost a year nobody had. Almost all of the thinking goes into getting in, which is entirely reasonable, because nobody selects a platform expecting to leave it and with luck nobody ever does.

Then something moves that was never part of the evaluation. Regulation. Ownership. Pricing. Geopolitics. Occasionally just a change of strategy.

And the organization finds out what it actually built: proprietary interfaces, data in formats nothing else reads, licensing written on the assumption of permanence, services coupled

tightly enough that they cannot be taken apart, and several years of accumulated data that now has to go somewhere.

None of that was visible while everything was working, which is the whole point of it. Cloud and sovereignty questions get discussed as though they were about where data sits. In my experience they are mostly about this: how difficult somebody has made the next change, and whether anyone knew they were deciding that at the time.

The cost of a dependency is paid at the moment you need to change it, which is reliably the moment you are least able to absorb it.

And the cost is paid by the people furthest from the reason

This is the part I find most worth noticing, because it explains why the pattern survives.

The team handling sensitive customer data lives with the controls and understands exactly why they exist. They are not the ones complaining. The friction lands hardest on the team doing something small and low-risk, who experience the whole apparatus as inexplicable — and who have no standing to question it, because questioning a security control from the outside looks like not taking security seriously.

So the people who could most easily say *this does not apply to us* are the people least able to say it, and the people who could change the rule rarely feel the cost of it.

Nobody chose to run the whole organization at the speed of its most cautious corner. It is simply what happens when a rule is written once and nobody is responsible for asking where it belongs.

Cultivating this: Different Capabilities, Different Conditions and When Shared Infrastructure Becomes Shared Friction, in Part III.

CHAPTER 33 · TECHNOLOGY

Faster, and Still Wrong

Technology multiplies a capability. It has no opinion on whether the capability was any good.

REDUCES FRICTION · WORKS ON ALL FOUR

Every organization has bought a tool to solve a problem that wasn't a tool problem.

A collaboration platform to fix communication between two departments that don't agree who owns the outcome. A project tool to fix delivery that was actually blocked on a decision nobody had authority to make. A data platform to fix reporting when the reports were already accurate and simply arriving too late.

None of those purchases fail, exactly. The tool gets installed, adopted, and reported as delivered. The friction stays where it was, now with better dashboards pointed at it.

Technology scales a capability; what it cannot scale is the judgement about when and whether that capability should be used at all.

That distinction does a lot of work. A tool takes something the organization can already do and lets it do it more, faster, or at lower cost per unit. That's genuinely valuable and it's most of what technology is for. What a tool cannot do is supply the capability in the first place, or decide when the capability should be used differently.

So the useful question before any significant technology decision is not *what will this let us do*. It's *what capability does this scale, and is that capability currently good enough to be worth multiplying*.

Scale a capability that isn't working and you get more of what wasn't working, arriving faster, in more places at once. This is a recognizable pattern in any organization that has automated a bad process: nobody made anything worse, and everything got harder.

There's a second thing worth naming, because it's where technology creates friction rather than removing it. Every tool encodes assumptions about who decides what. A workflow system with an approval step has made an ownership decision, whether or not anyone at the time thought of it that way. A budgeting tool that only accepts annual figures has decided your funding rhythm.

Those decisions are architecture, made by a vendor, adopted by procurement, and discovered by everyone else two years later.

The practical version of this is a decision you can see. ThoughtWorks publishes its Technology Radar twice a year — tools, techniques, platforms and languages each placed as adopt, trial, assess or hold, with a stated reason — and the organizations I've seen get value from it are not the ones that read theirs. They are the ones that build their own. What the exercise produces is not the list. It is a standing forum where *what does this multiply* has to be answered out loud, before procurement rather than after it, and where **hold** is a legitimate and recorded outcome rather than a failure to decide.

Technology multiplies whatever is already there. Which makes it a poor first move and an excellent second one — after the capability is clear and the ownership is unmistakable.

A note on engineering practice

There is a whole layer this paper deliberately does not cover, and leaving it unnamed would be a kind of dishonesty.

Between the conditions I have spent these chapters on and the value that eventually reaches someone, there is a set of engineering practices that determine how quickly and safely software actually moves: continuous integration, trunk-based development, automated testing, deployment automation, small batch sizes, the ability to release without ceremony. In a software-driven organization these are not incidental. They are where a great deal of friction is either removed or manufactured, day by day.

I have not written about them here for two reasons, and neither is that they don't matter.

The first is that they are already well covered, and covered better than I could. Anyone wanting to improve delivery practice should start with DORA's programme, summarized in *Accelerate*, rather than with me — the sample sizes and statistical work are beyond anything in this paper.

But I want to claim something from that research rather than only point at it, because it is the strongest external evidence this paper's central argument has.

What DORA established, across years and tens of thousands of respondents, is that the practices which reduce friction in the delivery path — small batches, automated testing, deployment without ceremony, loosely coupled architecture, teams that can release without asking permission outside themselves — produce measurably better organizational performance. Not better developer satisfaction, though that too. Commercial performance: profitability, market share, productivity.

Read that as a claim about friction rather than a claim about engineering and it says something considerably larger than it is usually taken to say. **Reducing structural friction increases throughput, and increased throughput is measurable value.** The mechanism is not motivational and it is not cultural. Work that used to wait stops waiting.

Two of DORA's four measures are, in this paper's vocabulary, friction measures wearing engineering clothes. Lead time for changes is decision latency and handoff loss combined, measured on the delivery path. Deployment frequency is what happens to throughput when both come down. The other two — change failure rate and time to restore — are the check that the speed was real rather than borrowed, which is the same distinction this paper draws between value grown and value extracted.

That is one domain, measured properly. The argument of this paper is that the same relationship holds outside the delivery path — in decisions, in ownership, in the boundaries between teams — where nobody has instrumented it and the friction is consequently larger. I cannot prove that at DORA's standard of evidence, and I am not going to pretend otherwise. But the one place it has been measured properly, it held.

The second is that they sit downstream of the conditions this paper is about. A team that cannot deploy without a release board is rarely blocked by tooling. It is blocked by ownership that never reached it, by a guardrail nobody settled, by a decision that has to travel. Continuous delivery in an organization with unclear ownership produces faster delivery of things nobody owns — which is the same trap as scaling a broken capability, arriving one layer down.

So the honest position is that these are two halves of the same problem. The practices are how flow is realized in software. The conditions are what determines whether the practices survive contact with the organization. Improve the practices and the conditions will limit how far you get. Improve the conditions and nothing happens at all until the practices catch up.

Technology multiplies a capability and has no opinion about whether the capability was worth multiplying.

What moves this is sequence rather than restraint. Establish who could say the output was wrong before scaling anything; ask what a platform assumes about how work gets approved before asking what it does; and pair one movement measure with one outcome measure so speed cannot be mistaken for progress. Nobody has to give up the platform — the argument is only about the eight weeks before it arrives.

Cultivating this: Multiplying What's There, in Part III.

If your delivery is slow and your ownership is clear, this paper has little to offer and *Accelerate* has a great deal. If your delivery is technically excellent and things still take nine months, the problem was never in the pipeline.

What AI Lands In

The same friction, amplified — and far less forgiving of the conditions it arrives into.

BOTH · WORKS ON ALL FOUR

I have now watched several organizations run what is essentially the same AI pilot. Comparable technology, comparable use case, comparable enthusiasm in the room on day one.

In one of them it was in production within months, and the interesting part was how unremarkable that felt to the people involved. They already knew who owned the process it touched, so they knew who got to say it was good enough.

In another it produced a genuinely impressive demo, followed by about eighteen months of conversation about data ownership, most of it perfectly reasonable, none of it resolving, because the question of who owned that data had been unsettled long before anyone thought of using AI on it.

In a third, nobody could establish who was allowed to decide whether the output was accurate enough to act on. Not because people were being difficult — because the work in question had always been reviewed by three different functions, and now something had produced an answer that all three had opinions about and none owned.

Same technology, three outcomes, and the technology explains none of it.

Everything in the previous chapter is true of AI, more so, and faster.

That is the whole of the sober case, and it is worth stating plainly before anything else, because AI is currently discussed as though it were a different category of thing rather

than the same category operating at a different speed.

AI amplifies organizations. It doesn't repair them. It reveals them.

It is worth being plain about what the word covers, because the public conversation carries a great deal of inherited fiction.

Despite the name, today's AI is not intelligence in the human sense. It has no goals of its own. No intent. No understanding of why anything matters. It is an extraordinarily capable prediction system, generating outputs from patterns, instructions and context — and being extraordinarily capable is not the same as being autonomous.

That distinction is not a way of talking AI down. It is the reason the rest of this chapter holds. The AI of science fiction pursues its own objectives; today's AI executes ours. A system that executes our objectives inherits whatever clarity we had when we set them, which is exactly why it amplifies an organization rather than replacing one.

An organization with clear ownership, capabilities it can name, and decisions made close to the work will get compounding value from AI, because it has something coherent to amplify. An organization where nobody can say who owns what will get faster production of things nobody owns.

AI also changes where complexity lives. Every major reduction in technical friction increases the importance of organizational architecture.

For decades, building software was constrained by scarce technical expertise. Ideas waited for engineering capacity. Improvements competed for developer time. Organizations naturally optimized around that constraint.

AI is rapidly changing it. More people can now describe software instead of writing it. More ideas become working systems. More experiments become products. Technical friction

is falling at a pace few organizations have experienced before.

But removing technical friction does not remove complexity; it relocates it — into ownership, into interfaces, into information, into decisions, into the architecture of the organization itself. The friction changes form rather than going away, and the form it takes next is structural. What gets removed is the technical part. What gets exposed is everything that was sitting underneath it, previously hidden by the wait.

Every unclear ownership, every unnecessary dependency, every weak interface and every ambiguous decision becomes more visible when software can be created in hours instead of months.

Organizations rarely become chaotic because AI is fast. They become chaotic because their structures were already struggling to absorb change.

There's a more specific version of this that matters for how you organize. An AI system performs best under the same conditions a team does: a clear purpose, explicit boundaries, and a well-defined responsibility. Ambiguity doesn't get smarter because you automated it — it produces confident output built on an unclear question, which is considerably harder to catch than a person saying "I'm not sure what you're asking."

Which brings the chapter to the question being asked badly almost everywhere.

The public conversation is about what AI will replace. I think that's the wrong question, and it's the wrong question in an interesting way. Whether AI can *do* the work is increasingly settled. Plenty of it, it can, and the remaining argument is mostly about timing.

The question that isn't settled is who owns the outcome.

A system can execute a decision. It cannot be accountable for having made it, because accountability requires someone who can answer for the outcome, explain the reasoning, absorb the consequence, and decide differently next time.

And ownership is a great deal more than approval, which is where this argument usually gets flattened.

Ownership is judgement. It is deciding what matters, and what matters more. It is weighing outcomes that cannot both be had. It is holding the context that made a decision reasonable at the time. It is accepting responsibility when it turns out not to have been. It is explaining the reasoning to someone who has to live with it. It is learning from the consequence, and choosing differently next time.

AI can assist every one of those activities, and increasingly does a respectable job of it. What it cannot do is hold them.

AI amplifies human capability. It does not take over the judgement, responsibility, accountability and ownership that capability gets exercised within.

That distinction becomes more important as AI systems become more capable, not less. The more decisions AI can support, the more valuable human judgement becomes — not because people can outperform AI at every task, but because an organization still needs someone who owns the outcome.

Execution can be delegated. Ownership cannot.

That isn't a sentimental distinction, and it isn't a claim about what AI will eventually be able to do. It's structural. An organization that quietly moves its people into execution-only roles while AI handles the deciding hasn't become more efficient — it has removed the part of itself that was supposed to own anything, and it will discover this the first time something goes wrong and there is no one who can say why the decision was made.

The organizations that will do well with AI are not the ones that adopt it fastest. They're the ones where ownership was already unmistakable, so there was something for the amplification to attach to.

The boundary this dissolves

Everything above is the defensive case. There is a second claim, it is the one I actually care about, and it is the reason this chapter sits where it does rather than at the end of the paper as a topical afterthought.

Building software is on the same path financial modelling took: from a specialist activity you queued for, to something people simply do as part of their work. Financial modelling was once specialist enough that organizations queued for experts; today there is no spreadsheet department and no spreadsheet request process. The competence did not disappear — it dissolved into the work itself. AI is the mechanism that finishes that journey. Not an accelerant on an existing trend — the thing that removes the last structural reason for the boundary to exist.

Consider what the boundary between IT and the business was ever *for*. Governance, standards and security are serious disciplines with real stakes, and none of them explain why the boundary exists in the first place — they are what grew up around it, not what caused it.

The boundary exists because turning an intention into a working system required a scarce specialist competence, and scarce competences get gathered into a function, given a queue, and managed as a supply. Every artifact of that arrangement — the requirements document, the intake process, the prioritization forum, the delivery contract, the phrase *the business* used by people who work for the same company — descends from a single premise: that someone has to translate.

That premise is the thing that is going.

Not entirely, and not evenly. Complex systems, integration, data at scale, and the parts where being wrong is expensive remain specialist work for a long time yet. But the large middle — the ordinary building that consumes most of the capacity of most technology functions — is moving to the people who know why the thing is needed. Not eventually. It has already started.

The consequence is profound. When software stops being scarce, organizations will create far more of it. More workflows. More automations. More AI agents. More experiments. More local solutions.

Architecture therefore becomes more important, not less. Not because more control is required, but because more creation requires better coordination.

Which means the question in front of every technology leader is not how to adopt AI. It is what their function is for once translation stops being scarce.

I think there is a good answer, and it is not a smaller version of the same thing. What survives is the part that was never really about translation: guardrails, interfaces, the discipline of ownership, and the judgement about which decisions are cheap to reverse. In other words, the competence this paper has spent Part III describing — moving into the organization rather than sitting beside it. IT not as a better partner to the business. IT as something the organization simply knows how to do.

In many ways, architecture becomes less about technology than it has ever been. Its role is no longer to design systems that people cannot build themselves. Its role is to design the conditions in which thousands of independently created systems can still behave like one organization.

I should be as honest here as I was in Part II: this is a direction I am confident about and a timeline I am not. An organization that misjudges the date loses a year. One that misjudges the direction spends that year defending a boundary that has already gone.

And the condition attached to it is the same condition, only stricter. Distribute the ability to build without distributing the discipline of ownership and AI does not dissolve the boundary. It floods the organization with systems nobody owns, built by people who were never told what good looks like, at a speed no governance process was designed to absorb. The spreadsheet problem, at machine scale.

There is an irony in all of this. For decades we assumed technology was the difficult part. AI suggests it may turn out to be the easiest, while coordination, shared understanding and ownership stay about as hard as they have always been — none of them has a version that arrives in a release note.

Organizations that mistake easier software for easier organizations will get more software, and more friction along with it. The ones that also work on the structural side get something different, and it is worth being precise about what: not building faster, but a larger share of what they build actually reaching somebody outside.

AI is the most powerful multiplier currently available and the least forgiving of the conditions it lands in. That is what makes the rest of this paper more urgent rather than less relevant.

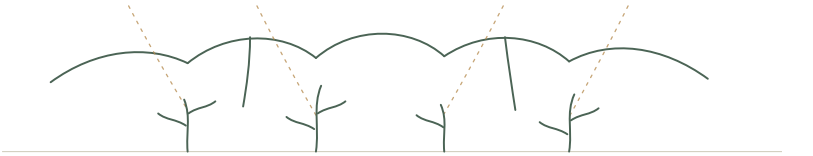
AI amplifies the conditions it lands in, which is the whole of the sober case and most of the practical one.

What moves this is checking ownership before capability — taking three things AI would plausibly accelerate and asking who owns the outcome of each, by name. Where that returns a name, acceleration is straightforwardly good. Where it returns a committee — which is an owner, just a slower one with the accountability shared out — the question becomes

whether it can answer as often as it is about to be asked. The two weeks spent settling that first are cheaper than the pilot and very much cheaper than the eighteen-month conversation.

Cultivating this: Owning What AI Can't, in Part III.

As technical friction continues to fall, Organizational Flow becomes the primary determinant of how much value an organization can actually realize. Which makes the question of how it is deliberately cultivated the only one left — and that is the whole of Part III.



PART III

Cultivating Organizational Flow

How do the conditions behind those patterns get cultivated?

Each chapter picks up one of the patterns from Part II and asks what can actually be tended around it. Not a fix — conditions, which drift back the moment nobody is watching them.

Very little of this looks like a programme. Most of it is small, unglamorous, and available to somebody on a Tuesday without asking anyone for permission first.

The last two step back and take both halves whole: friction, which is cultivated by taking things away, and value, which is cultivated by contact.

CHAPTER 35 · CULTIVATION

Tending What Grew

You cannot redesign a structure nobody designed. You can prune one.

BOTH · WORKS ON OWNERSHIP CLARITY

Two observations in Part II end without anywhere obvious to go. It Grew Like That says the shape arrived one reasonable decision at a time and has no author to consult. The Conditions Don't Come in the Box says the borrowed shape will not bring what made it work.

Between them they rule out the two things organizations normally do: redesign it, or copy somebody else's. Which leaves the question this chapter is for — what do you actually do with a structure that grew?

You tend it. Which is slower, less impressive, and the only thing I have seen work.

Start with what it makes hard, not with what it is

The instinct is to map the current structure and look for what is wrong with it. That produces an accurate diagram and no decisions, because there is nothing wrong with it — every part of it made sense to somebody at some point.

The more useful question is narrower. Take the three things that most recently took far longer than they should have, and for each one, trace where it waited. Not who was slow. Where it sat.

You will get a small number of places, repeatedly. That set is your actual structure — the load-bearing shape, as opposed to the drawn one — and it is usually four or five boundaries rather than the forty on the chart.

This works because friction is honest in a way org charts are not. The chart tells you what somebody intended. The waiting tells you what exists.

Prune one thing, then wait

The strong temptation, once you can see it, is to fix all five at once and call it a reorganization. Which converts tending back into redesign, and reintroduces every problem Part II described — including that nobody afterwards will be able to say why the new shape is the way it is.

One at a time is not caution, it is information. A grown structure has dependencies nobody has documented, and the only reliable way to find them is to change one thing and see what complains. Change five and you learn nothing about any of them.

Wait long enough to see the second-order effect, which is usually a quarter. Things get briefly worse first — that is what removing a compensation feels like from inside, and it is the point at which most attempts get reversed.

Expect less resistance than you have budgeted for

This is the part that surprises people, and it follows directly from the observation.

Nobody chose the current arrangement, so nobody is defending it. There is no author whose reputation is attached, no original decision anyone is protecting. What looks like resis-

tance is nearly always something narrower: a person who has built a working accommodation around the awkward boundary and is not confident the replacement will work as well.

That is a solvable conversation, and it is a completely different conversation from the one people prepare for. Most of the political energy spent on structural change is spent arguing with an opponent who is not there.

Borrow questions, never shapes

The Conditions Don't Come in the Box argues that the transferable part is the question somebody asked, not the structure it produced.

So when something works somewhere else, the useful exercise is archaeological: what problem was this solving, and do we have that problem? Usually the answer is *a version of it, differently shaped* — which is far more useful than the model, because it sends you looking at your own conditions rather than at their diagram.

The failure mode to watch for in yourself is the one I know best. You come back with something good, and you want to give people the answer rather than the question, because the answer is the part that took you three days at a conference and the question sounds too simple to be worth flying anywhere for.

Keep a record of why, because the next generation won't have one

Everything above produces a structure that grew — just more deliberately. Which will be equally opaque in eight years unless somebody writes down not the shape, but the reason.

One paragraph per boundary change: what was waiting, what we moved, what we expected. That is enough for whoever inherits it to know whether the reason still holds, which is the only question that matters and the one no org chart has ever answered.

A structure nobody can explain will be reorganized eventually, by somebody who cannot see what it was for. Writing down the why is the cheapest defence against your own successor.

This cultivates: It Grew Like That and The Conditions Don't Come in the Box, in Part II.

CHAPTER 36 · CONDITIONS

Growing the Team

Cultivating the conditions where good work keeps happening.

CREATES VALUE · WORKS ON OWNERSHIP CLARITY

The cultivation idea first came up in a conversation with my friend and colleague Lars Barkman. We kept turning it over, and the longer we did, the clearer something became that I'd spent years circling without ever saying plainly:

> **The work isn't designing solutions. It's cultivating the conditions where good solutions keep emerging.**

That took a while to admit, because designing solutions is the part of the job that feels like the job. It's visible. It produces something you can present. Cultivating conditions produces, on a good day, nothing you can point at — just an organization that turned out not to need you in the room.

And it isn't a metaphor. It's the thing the metaphor had been pointing at all along.

You cannot specify a tree into existence, hand it a deadline, and expect fruit by Friday. You can only create the conditions — soil, light, water, room — and then do the patient, unglamorous work of tending while the tree does what trees do.

Teams are the same. You cannot manage a team into greatness. You can only cultivate the conditions that make greatness possible, and get out of the way often enough to let it happen.

That distinction — between managing and cultivating — is most of what this chapter is about.

This isn't a software observation. An operating theatre team, a newsroom on deadline, a restaurant kitchen at service — each is a small group with a shared outcome, real interdependence, and no time to route decisions upward. They work well or badly for almost exactly the reasons software teams do, which is a strong hint that the pattern isn't about the work at all.

A healthy root system is never uniform. Some roots drive deep for water, others spread wide and shallow for what only the topsoil holds, and the fine ones nobody ever sees do most of the actual work. A team works the same way: different competences reaching into different parts of the problem, all drawing on the same ground. The best moments happen when that variety moves in sync — covering for each other, playing to each other's strengths, growing toward the same light.

Nothing about cultivating is passive, though. Anyone tending a tree still has a plan — which limbs to let run, which to cut back, what needs staking before it splits under its own weight. Teams need the equivalent: guiding principles that connect each day's work to something larger, so a sprint isn't just busy, it's aimed.

Without that plan you don't get a strong tree. You get scrub.

Underneath all of it is soil, and nothing grows in soil compacted by fear. Trust improves the way soil does — slowly, through repeated seasons of people knowing what to expect from each other. Clarity about vision, goals and responsibility is what keeps the ground loose enough for roots to spread. A culture of open communication and tolerated risk isn't something you add on top of trust. It *is* how trust gets built, one honest conversation at a time.

Three conditions show up again and again in Daniel Pink's work on what makes people thrive — autonomy, mastery and purpose — and they map onto cultivation almost too neatly. Autonomy is room to grow in your own direction within the ground you've been given. Mastery is the slow thickening of a root system that only comes from staying in one place long enough to actually take hold. Purpose is what the whole tree is reaching for.

Nobody plants without deciding where one stand ends and the next begins — not out of suspicion, but because clear edges are what let each stand be tended on its own terms, without one plot's problems choking the plot beside it.

Organizing teams around areas of the business is the same instinct. And how you draw those boundaries matters more than most organizations realize, because the shape of the teams becomes the shape of what they produce, whether anyone planned for it or not. Rigid, centralized communication produces rigid, centralized results. Teams that can act independently produce things that can change independently.

The layout and the yield are never really two separate things — which is the introduction's arithmetic restated at the scale where you can actually do something about it. You don't get an adaptable result out of an organization that has to convene to agree on anything.

The healthiest stands, in the end, aren't the largest. They're the ones sized so one person can reach every tree without trampling three others to get there. Teams work the same way: small enough that everyone can still see the whole plot, organized around a domain they actually own, given autonomy, mastery and purpose, and cultivated with clarity and trust rather than instruction.

Where exactly that line goes, and who gets to draw it, is its own piece of work — Drawing the Line Around a Team is the chapter for it.

The friction this removes has a specific shape: the waiting, the checking, the asking that fills the space where conditions were never shaped. A team that has to ask permission for every judgement call hasn't been undertrained. It's been under-cultivated — nobody ever settled the ground it was meant to grow in.

None of which tells a team what it is for. Conditions make good work possible; they don't say what work. And the answer to that has to be something more durable than the project currently in front of them, or the conditions will be rebuilt from scratch every time the project ends.

A system that needs cultivating, and a way of cultivating it. That's the baseline underneath everything this paper calls Organizational Flow.

Who you hire is part of the cultivation

Conditions are not only shaped by what you settle. They are shaped by who is standing inside them — which organizations almost never look at this way, because hiring is filed under recruitment and friction is filed under process, and the two folders are rarely opened in the same meeting.

The advice everyone gives is *hire the best people*. It sounds unarguable and it is nearly useless, because it contains no information about your organization. Best at what, arranged how, deciding what on their own authority.

I once helped recruit someone genuinely excellent into a structure that could not use them. Entrepreneurial, fast, used to deciding and then telling people afterwards — exactly the profile the hiring brief asked for, and I remember being pleased with the appointment. They arrived into an organization where significant decisions went to a forum that met fortnightly, and where deciding first and explaining later was read as a governance problem rather than as initiative.

Within six months they had learned to wait. Within a year they had gone, and the exit conversation was polite and entirely about pace.

Nobody had done anything wrong. The person was excellent, the structure was coherent, and the combination produced friction that neither party had signed up for. What was missing was anyone asking whether the organization we actually had could use the person we were actually hiring.

So the real question is not who is best. It is who will move well through the organization you actually have — or through the one you have decided to build, if you have decided.

The failure runs in both directions, and both are common.

Entrepreneurial people hired into a control structure. They decide, get corrected, decide again, get corrected again, and then stop. What follows is worse than if you had hired someone compliant, because you have paid a premium for judgement and then trained it out. The organization concludes that the hire did not work out. The hire concludes the organization was not serious, and both are describing the same structural mismatch from different sides.

People uncomfortable with responsibility hired into distributed ownership. This one is quieter and takes longer to see. Ownership gets assigned and nothing is decided. Questions travel upward that were meant to stop where they started. Leaders conclude that autonomy does not work here, and quietly recentralize — and the recentralization is now supported by evidence, which makes it very hard to argue with. What actually happened is that the organization staffed a structure with people who never wanted it.

Neither profile is better. Both are entirely reasonable people who would thrive somewhere else in the same market.

Which makes hiring one of the slowest forms of friction to undo. A team boundary can be redrawn in an afternoon. A decision right can be moved with a memo and some follow-through. A mismatch between a person and the structure they work inside lasts as long as they stay, generates friction every week of it, and is nobody's to fix — because it does not look like friction. It looks like a person.

Three things follow, and none of them require a programme.

Decide the organization first, then hire into it. If you want distributed ownership, say so in the room, and hire people who visibly want to own something. If you want centralized control — which is a legitimate choice for some businesses, and the paper's argument does not make it illegitimate — then hire people who work well inside it, and stop recruiting on the promise of an autonomy you do not intend to grant.

Interview for the conditions, not only for the craft. The most useful question I know is what someone did the last time they disagreed with a decision that had already been made. The answer tells you where they will sit in your structure far more reliably than a technical assessment does.

Be honest in the recruiting conversation about how decisions actually get made here. Not how the operating model says they get made. The honest version costs you some candidates and saves you the ones who would have left in a year anyway.

The organizations that get this right are not the ones with the strongest employer brand. They are the ones that knew what they were building and hired people who wanted to build that.

Keep the competence current, because it is depreciating anyway

The Half-Life of Knowledge argues that professional knowledge loses relevance whether or not anybody notices, and that the loss is not only technical — how customers behave, what leadership means across three time zones, what a business model can be now all move just as much and much more quietly.

A team's competence is the thing standing behind whatever capability it owns, so this is a condition like any other, and it is one of the few that decays on its own if left alone.

Treat time spent learning as work. Not as a benefit, not as something generously permitted, and certainly not as a hobby subsidized by evenings. The organizations where competence stays current are the ones where a Wednesday afternoon spent on something not yet useful does not need defending, and the tell is whether anybody feels the need to explain what they were doing.

Notice who has not learned anything uncomfortable lately. Including yourself. The failure mode is not somebody who has stopped caring; it is somebody excellent whose picture of the world was assembled a few years ago and has been performing beautifully ever since against a version of the problem that has moved.

Let the two-year person teach the twenty-year person something. Everybody is partly out of date, in different places, and a team that can say so out loud is considerably faster than one where seniority is treated as a proxy for being right. This is quietly the most levelling thing about the pace of change, and it only pays off if somebody makes it normal.

And the individual half is worth saying plainly, because no organization can do it for you: nobody else is going to own this on your behalf.

Notice the loud one before they go quiet

It Shows Up in People First describes the sequence: frustration, then silence, and usually the same person. The practice is catching the first phase, and it is mostly a matter of what you do in the ten seconds after somebody says something sharply.

Ask a frustrated person what they were trying to reach. Not to calm them down — they will hear that instantly and it will cost you the next three months of honesty. Ask because they have almost certainly identified something real and are the only one still willing to say it out loud, and because the tone that is bothering everyone is the last available evidence that they still care.

Ask a quiet person what they stopped suggesting. Expect to ask more than once, and expect the first answer to be polite and untrue, because the honest answer is unflattering to whoever is asking. This one is much harder to act on than the first, which is the whole argument for getting to people earlier.

And when somebody does raise something in the loud phase, the useful response is to do something visible about it within a fortnight, even something small. What produces the quiet version is not being disagreed with. It is being heard, agreed with, and then watching nothing happen.

Make both places work, then trust everybody

Where the Work Actually Happens ends on a position I would defend: the place is negotiable, the presence is not. What follows from that is less about buildings than people expect.

Make both modes genuinely possible. The tooling, the norms, the defaults — enough that someone choosing either one is not choosing a worse experience. Then let people choose against the task in front of them rather than against a day of the week.

Extend the trust to everyone, not selectively. Selective trust is worse than none, because it is visible. People know exactly who is being watched, and everyone else adjusts their behaviour to avoid becoming that person. What you get is performance of presence rather than presence itself. If somebody isn't contributing, that is a conversation with that person about contribution — which most organizations avoid by writing a policy instead. A policy is what you reach for when you would rather not have the conversation, and it costs everyone a little so that nobody has to.

Either everyone is in the room or everyone is on the call. Not half and half. This is the one rule here I would defend hardest, because the mixed meeting reliably drops the people on the screen and everybody genuinely believes they were included.

Cameras on, and I'll take the argument about it. Nobody in a physical meeting slides under the table or reads unrelated mail while somebody explains something to them. On a call it is routine, and a black rectangle makes it invisible. A camera is not surveillance; it is the minimum signal that you are actually there. I will concede that a laptop makes this genuinely hard — everything else you might be doing is one keystroke away, permanently — but that is a discipline problem rather than a reason to disappear, and the discipline is basic courtesy to whoever is currently presenting to an empty grid.

And when attention keeps drifting in the same recurring slot, ask what the forum is for. Not rhetorically. What decision does it make? What would fail if it stopped? Who put it there, and are they still here? Sometimes there is a good answer and you should show up properly. Often enough there

isn't, and the honest version is that it was set up to solve something since solved, and kept running because a recurring invitation is one of the most durable objects in any organization. Challenging it is whoever noticed's job, and that is usually the person half-listening — which is an awkward place to raise it from, and still the right one.

Great teams are grown, not assembled. They need space, light and time — and someone patient enough to keep tending them after the planting is done.

Which is the pattern for everything that follows. Each chapter in this part takes one of the conditions Part II described and asks the same question: not what it is, but how it is deliberately strengthened, by someone, over time.

This cultivates: Downstream of the Conditions, It Shows Up in People First, The Half-Life of Knowledge and Where the Work Actually Happens, in Part II.

EDITOR'S NOTES

Autonomy, mastery and purpose is Daniel Pink's synthesis in *Drive*; the underlying research is Edward Deci and Richard Ryan's Self-Determination Theory. Pink popularized it, he didn't discover it, and the difference matters if you want the evidence rather than the story. What this chapter adds is the reframe: treating those three as growing conditions to be cultivated rather than motivational levers to be pulled.

CHAPTER 37 · TEAM BOUNDARIES

Drawing the Line Around a Team

Where the boundary goes decides how much a team has to hold, how often it has to ask, and whether anybody chose any of it.

BOTH · WORKS ON HANDOFF LOSS

How Much Can One Team Hold makes the case that headcount is the wrong number and cognitive load is the right one. The Team That Has to Ask shows what it costs when the judgement a team needs sits on the other side of the line. This is the chapter about drawing that line deliberately, which is cheaper than almost anything else in Part III and gets done less often.

Count what they have to understand, not who they are

The move is unglamorous. Sit with a team and list the separate things they have to understand well enough to be trusted with — not the systems they touch, the domains they have to reason about. Four unrelated ones in a team of five is a heavier team than one coherent domain in a team of nine.

Then look for the one that does not belong. There usually is one, and it usually arrived because somebody needed a home for it at short notice and this team had a person who knew about it.

Moving it out costs a negotiation and no headcount. I have watched that produce a visibly faster team inside a month, and watched the alternative — adding two people to an over-

loaded team — make the load worse, because now there was more context to keep in sync and the four unrelated domains were still sitting there.

A capability is the boundary worth reaching for, because a capability is coherent by construction. The things inside it belong together, so understanding one helps with the next. Draw the line around a system, a technology or a reporting line instead, and the team inherits whatever collection of unrelated concerns happened to land in that box.

Put the judgement inside the line

This is the one I would reach for first, and it is a staffing decision rather than a process one.

Find the person the team keeps waiting on — the one whose judgement is needed before anything can be called finished. Move them inside the team. Then let them decide without checking, which is the half that gets skipped and the half that matters.

It will cost somebody a person they consider too valuable to embed. That objection is genuine and usually wrong: the specialist spread across six teams is not being leveraged, they are being queued, and the leverage everyone believes they are getting is mostly other people waiting politely.

Nothing else I have tried converts a recurring handover into a conversation across a desk. And when the competence genuinely cannot be embedded — there are three of them in the company and eleven teams need them — the honest answer is not a better queue. It is to look at what they are actually being asked for, and how much of it could be settled once, written down, and stop being a question.

Choose the interaction rather than inheriting it

Teams are not all the same kind of team, and the way two of them work together is a design choice that mostly gets made by accident — inherited from whatever the first integration happened to do and then treated as the nature of the relationship.

Close collaboration is expensive and should generally be temporary, with a stated end. Consuming something as a service is cheap and should be the default the moment the thing is stable enough to have an interface at all.

In this paper's terms that is a statement about friction rather than about org design. Collaboration is a handover you have chosen to keep, usually for a good reason. A service is a handover you have removed.

So the useful review is short: for each pair of teams that work together regularly, ask which of the two this is, whether anybody chose it, and — if it is collaboration — what would have to be true for it to end.

Let people place themselves, when the ground can take it

The best version of this I have been part of took less than two days. Everyone in a room, one rule — every product needs a team with all the capabilities required to succeed — and no manager deciding who went where. What they optimized for was the products, not themselves. Every team left knowing why it existed and what it owned, and nobody could say management put them there.

I would recommend it, with one caution I would rather give than leave out: the self-selection was not the active ingredient. Run the same exercise where the last three reorganizations were announced rather than discussed, and you get a room waiting to be told the right answer, followed by a quiet renegotiation afterwards by whoever has the most political capital.

What made it work sat underneath. People were trusted visibly, in a way that would have been embarrassing to walk back. They understood the purpose well enough to trade off against it. And they built the structure together rather than being handed one and asked for comments.

Which suggests the honest sequence. If the conditions are there, self-selection is the cheapest way I know to convert them into ownership. If they are not, the exercise will tell you so within an hour, and that is worth knowing too.

The smaller version works in almost any organization and is worth doing more often: whenever a boundary is about to move, bring the people it will move into the room while the decision is still genuinely open. People rarely resist responsibility. They resist responsibility for decisions they never had the opportunity to influence.

What this does not fix

A well-drawn team boundary removes handovers. It does not, on its own, tell the team what it is for, and it does not survive the next reorganization unless somebody is tending it — which is why the boundary worth drawing is the one that matches something durable rather than something current.

Boundaries drawn around capabilities tend to survive. Boundaries drawn around projects, systems or the current leadership arrangement get redrawn with them, and every redraw spends the understanding the last one built up.

This cultivates: How Much Can One Team Hold, The Team That Has to Ask and Nobody Could Say Management Put Me Here, in Part II.

CHAPTER 38 · UNBLOCKING

Ten Minutes Today

Cultivating the one form of friction no structure can reach.

CREATES VALUE · WORKS ON DECISION LATENCY

The Wait Nobody Logged is the chapter where the structural argument runs out. You cannot draw a boundary that makes people answer each other faster, and Nobody Opts Out of the Arithmetic says why it matters: every verb in this paper still passes, at some point, through one person choosing to help another.

So this chapter is about the only lever there is, which is norms — and norms are slower to build than structure, harder to explain to a steering group, and considerably more durable once they hold.

Make unblocking somebody count as work

The reason a blocked colleague waits two days is almost never indifference. It is that answering them is not on anybody's list, and everything else is.

The norm worth establishing is narrow enough to state in a sentence: *a question from somebody who is blocked gets ten minutes today, not a proper answer next week.* Ten minutes is usually enough to unblock. The proper answer is usually the enemy — it is what makes the question feel like a project, which is what makes it wait.

What makes this stick is not announcing it. It is what happens in the first few weeks when somebody visibly drops what they are doing to unblock a person, and is thanked for

it in front of others, by somebody whose opinion carries. That is a five-second act and it does more than any policy on responsiveness.

The opposite also teaches, faster. An organization where the person who stopped to help is asked why their own deliverable slipped has settled the question, and will not need to discuss it again.

Notice who is waiting, because they will not tell you

A person waiting produces no signal. They have moved on to something else, which is the professional response and also the thing that removes the evidence.

So the noticing has to be deliberate, and the cheapest version is a question with a name in it. Not *is anyone blocked* — which reliably returns silence, because admitting you are blocked sounds like admitting you are stuck. Instead: *who are you waiting on, and since when?*

That question is answerable, unembarrassing, and produces a list of names and dates within about a minute. It is also the single most useful thing I know to ask a team, and the reason it works is that waiting on somebody is not a failure, so nobody has to defend it.

Then ask the harder half, of yourself: *who is waiting on me?* Expect the answer to be longer than you think. Everyone underestimates this one, including me, and it is the only version of the question that changes your own behaviour rather than somebody else's.

Then notice which questions should not have needed asking

There is a version of this chapter that would do harm, and it is worth naming plainly: an organization where everybody is admirably responsive to questions nobody should have had to ask.

Ten minutes given generously is still ten minutes, and a question answered warmly is still a handover. An Interface Nobody Can Find puts the test in one line — if the request lands in somebody's list, it is a queue, however well the queue is run — and You Have to Know a Guy shows where a well-run queue ends up when the queue is a person: a capability reachable only through somebody who is being thanked for being the route, right up until the day they leave.

So the norm needs a companion, and the ordering matters. Answer today. Then, if the same question arrives a third time, stop answering it and write the answer down where the next person will find it without you.

That second half is what separates generosity from load-bearing generosity. The first is a good colleague. The second is a single point of failure with excellent manners, and the organization will call it indispensability and mean it as praise.

The order worth holding is short. Automate it if it does not need a human at all. Failing that, make it self-service, so somebody can get what they need without entering anyone's list. Failing that, run a proper queue with a name on it. And only when none of those apply does it belong in somebody's ten minutes.

Which rung a thing sits on is a statement about where the organization has got to, not about how careful anybody is. A team that reviews something by hand because nothing yet checks it automatically is behaving correctly. The friction is

real and so is the reason for it, and removing the check before building its replacement would be worse than leaving it alone.

What that position needs is a date rather than a defence. The rung you are on is a decision about what to build next, and the failure mode is the sentence *we are not ready for that yet* — which is true when first said, needs nobody to renew it, and can hold a manual step in place for six years while everyone involved agrees it is temporary. Ask about the ladder every so often and the answer stays honest. Never ask, and the current rung stops being a stage and becomes a description of the organization.

Most of this chapter is about the bottom of that list, because it is where the residue collects. Opening the Front Door is about moving things up it. The two are not in tension: the questions that genuinely need a person are precisely the ones worth having your afternoon, and there are far fewer of them than the current volume suggests.

Trust is the compound interest here

Trust does not grow from stated values and it does not arrive at an offsite. It grows when people repeatedly help each other move: the same small transaction, often enough that it stops being a pleasant surprise and becomes a reasonable expectation.

That is a slow accumulation and a fast loss, in that order. Which is worth knowing before you start, because the payoff schedule is unhelpful — months of small unremarkable helpfulness, then a step change in how quickly things move that nobody can attribute to anything.

Nothing in this chapter is measurable, and I would treat anyone offering you a metric for it with suspicion. What you can observe is the second-order effect: when the norm holds,

questions get smaller. People ask sooner, because asking is cheap, which means they ask before they have spent two days going the wrong way.

Where it breaks

It breaks under load, and it breaks from the top.

Under load, because ten minutes for somebody else is the first thing cut when your own week is full — and a norm that only holds in quiet quarters is not a norm, it is a mood. The organizations where this survives are the ones where somebody senior is visibly still doing it in the bad week, which is exactly when it is hardest and most watched.

And from the top, because everybody calibrates on what the busiest, most senior person does with an unexpected question. If the answer is *forwards it*, that is the norm, whatever anybody has said.

Nothing shortens the queue in everybody's inbox except a few hundred people each deciding, repeatedly, that somebody else's afternoon is worth ten minutes of theirs.

This cultivates: The Wait Nobody Logged and Nobody Opts Out of the Arithmetic, in Part II.

CHAPTER 39 · LEADERSHIP

Leading Without Deciding

Doing the design work on purpose, in the rooms that never thought of themselves as designing anything.

BOTH · WORKS ON OWNERSHIP CLARITY

Decided Upstairs, Felt Downstairs made the case that the most consequential design work in an organization happens in rooms where nobody believes they are designing anything. Reorganizations. Budget rounds. The quiet choice to route one more decision upward because the last one went badly.

This chapter is about doing that work deliberately. None of it needs a mandate or a programme, and all of it requires giving something up, which is why it is harder than it looks and why it opens this part.

What changes

Structure stops being an HR artefact. A reorganization, a funding decision and a reporting line become what they actually are — choices about how much friction the organization will carry from now on, and how much of what it can do will reach anybody outside it.

Name what the design makes hard

Before a reorganization is announced, name three things it will make slower. There always are three. If nobody in the room can name them, the design has not been examined — it has been drawn.

I like this one because it is almost impossible to do defensively. Nobody has to admit the design is bad; they only have to say what it costs, which experienced people are usually quite happy to do once someone makes it acceptable. The conversation takes twenty minutes and I have never seen it fail to change something.

Fund capabilities for periods, not projects for dates

Project funding forces an end date onto something that has none, and produces a handover to nobody at the moment the outcome would have arrived. Funding a capability for two quarters costs the same money with none of the ceremony, and leaves someone still standing there when the results come in.

Funding What Doesn't End is the long version of this argument, including the part about which work genuinely should end.

Push one decision down and leave it there

Pick a decision that routinely reaches you, hand it to whoever already has the information, and then — this is the whole exercise — do not take it back the first time it goes badly.

It will go badly at some point. The retraction teaches far more than the delegation did, and what it teaches is that the delegation was conditional, which everyone will remember for years. I have done this wrong, and getting it back took considerably longer than it took to lose.

Watch what gets rewarded

No Incident, No Story makes the observation: rescue is legible, prevention is not, and an organization that celebrates the first reliably produces more things to rescue. This is what a leader actually changes about it, and it is narrower than it sounds.

Go looking, because nobody will come and tell you. Once a quarter, ask what did not happen — the incident that was designed out, the escalation that never arrived, the migration that was boring. The people who did those things will not raise their hand, partly from modesty and mostly because there is nothing to raise it about.

Then say the name in the same room where rescues get named. Not a separate mechanism. The same meeting, the same tone, the same weight. A parallel channel for quiet work reads as a consolation prize and everybody can tell.

And notice what you personally get animated about. A team calibrates on what makes the senior person in the room lean forward, and that signal is transmitted long before any policy is. If the crisis gets your attention and the uneventful quarter does not, you have already answered the question, whatever you have said.

The half that points outward

Everything above reduces friction. The version of this chapter I would have written ten years ago would have stopped there, and it would have been half a chapter.

The other half is a question worth putting into the same rooms: when this lands, who is better off, and how will we know?

It is an awkward question in a budget round, because the honest answer is often *we are not sure yet*, and budget rounds are not built for that. But it is the only question that distinguishes a structure designed to move quickly from a structure designed to move quickly toward something. Ask it early enough and it shapes the design. Ask it after the reorganization is announced and it reads as criticism.

The rooms this chapter is about are where both halves get decided, months before anyone downstream feels either of them. That is the whole reason for taking them seriously — and the reason friction is mostly created by people who would be genuinely surprised to hear they had created any.

This cultivates: Busy Toward Nothing in Particular, No Incident, No Story and Decided Upstairs, Felt Downstairs, in Part II.

CHAPTER 40 · ARCHITECTURE

Equipping, Not Reviewing

What changes when architecture stops being somewhere decisions go and becomes something the people deciding already have.

BOTH · WORKS ON DECISION LATENCY

A note on the word, for anyone dropping in here. This paper uses ***architecture*** *in a wider sense than the technical one — the deliberate shaping of how an organization is put together, of which system design is one part. A reorganization is an architectural decision. So is a funding model, a team boundary and a reporting line. What a Capability Is, and Is Not in Part I sets out the vocabulary this rests on.*

The Long Way to a Yes described decisions travelling to find the judgement they needed. This is the chapter about shortening that journey, and the honest summary of it is that you cannot. What you can do is move the judgement.

That sounds like a slogan, so here is what it looked like the first time I did it properly, some years after taking apart the board that taught me the lesson.

We had a recurring pattern where teams brought us integration questions. Reasonable ones — should this be synchronous, who owns the contract, what happens when the other end is down. Each conversation took about twenty minutes and each one had waited nine days for a slot. The nine days were not our fault and not theirs; that was simply how long it took two calendars to agree.

So we wrote down the answers. Not a standard, not a policy — four paragraphs about when to wait for a response and when not to, with the reasoning attached and a named person to argue with if it did not fit. It took an afternoon. Within

about two months the integration questions had mostly stopped coming, and the ones that did come were genuinely interesting, which was considerably more fun for everyone.

What that afternoon actually did was convert a queue into a capability. The judgement had been real and it had been ours, and it had also been sitting nine days away from every person who needed it.

What changes

Architecture stops being a place decisions are sent, and becomes a competence sitting where the decisions are made. That is a change in how people spend their week rather than in what anyone is called — and most of the people doing this work do not have the word architect in their title at all. In a lot of organizations it is an engineering manager, a head of product, a CTO, or whoever has been there longest and can still remember why the boundaries are where they are.

Making other people's decisions better

Every review is a decision that had to travel. That is not an argument against reviewing anything; some decisions genuinely need more eyes, and a handful are expensive enough to be worth the wait. It is an argument for knowing how many of them are in the second category. In my experience it is a small fraction of the total, and the rest are there because somebody was not trusted or, much more often, was not equipped.

The useful experiment is small: take one recurring review and try to convert it into something the team can carry. Sometimes that is a guardrail. Sometimes it is a conversation that only has to happen once. Sometimes you discover the review was doing something real, and you keep it — which is a good outcome too, because now you know.

Writing things down helps more than it should, on one condition: write down the decision and the reason, not the target state. The reason is what expires. Anyone who finds the record in three years needs to know what was true when it was made, so they can work out whether it still is. A target state tells them nothing except that somebody was once optimistic.

Being in the rooms that don't know they're architectural

Leadership's chapter made the case that the most consequential architecture gets decided in budget rounds, organization design, hiring and vendor selection, by people who would not describe any of it as architecture. The cultivation that follows from that is unglamorous: be in those rooms.

Not to review them. To be the person who says, quietly and early, *if we split it that way, everything that touches it needs both teams from now on*. That sentence takes four seconds and saves several years. It is also completely unpersuasive after the decision has been made, which is why the timing matters more than the argument.

The standard, and the other half of it

Once a proposal clears the floor that security and compliance set — and it is a floor, not a test — there is one question worth putting to it: does this leave the organization, and what it builds, easier to change than before.

That question is entirely about friction, and for a long time it was the only one I asked. It is not sufficient. An organization can become beautifully easy to change and change nothing

anyone wanted. The second question is slower and less comfortable: who is better off if this works, and how would we find out.

Most architecture conversations never reach the second one, because the first is answerable in the room and the second is not. But asking it changes the shape of the conversation immediately. Half the proposals I have seen survive the ease-of-change question fall apart quietly on *who is better off*, usually before anyone has had to say no — and the ones that survive both tend to be the ones people are still glad about two years later.

Governance asks whether a decision was approved. Architecture as a competence asks whether the people making these decisions are equipped to make them well, and whether anyone will be better off afterwards, and then does the slow, unheroic work of making both of those more true than they were last quarter.

This cultivates: The Long Way to a Yes and A Tollbooth on a Road With a Bypass, in Part II.

CHAPTER 41 · GUARDRAILS

Settling the Few Things

Cultivating the small set of decisions nobody should have to make twice.

BOTH · WORKS ON DECISION LATENCY

Handing a team a capability without also settling the principles they can lean on doesn't produce autonomy. It produces a team guessing at what leadership actually wants — slower and considerably more anxious than asking outright would have been, and harder to spot, because from the outside it looks like empowerment working.

I have done this to people, believing I was giving them room. What I was actually giving them was the job of reverse-engineering my preferences from whatever I had reacted to last. Guardrails are what make ownership safe to take.

They are not fences, whatever the word sounds like. A fence stops you from crossing a line. A guardrail tells you where the road is, so you can drive fast without needing anyone's permission to take the next turn. A team that knows the boundaries can take real risks inside them, challenge ideas, and recover from mistakes without the mistake becoming a crisis. Constraints, chosen well, don't reduce creativity. They give it somewhere to push against.

None of that removes the need for courage. It changes what courage costs — the difference between a call you can defend by pointing at something agreed, and one you are making entirely on your own credibility.

Which leaves the practical question. If architecture isn't a review board and isn't a folder of diagrams, what does it actually leave behind?

A short list of decisions everyone else no longer has to make.

That is what a guardrail is. Not a rule to be complied with — a question that has been settled once, deliberately, so a team starting something new inherits the answer instead of relitigating it on a Tuesday.

The first one on my list has nothing to do with technology, and I did not come up with it. I picked it up from the way AWS talked about theirs, and it fits in a sentence: do what is best for the company.

That reads like something printed on a wall and ignored, right up until you follow it properly. It holds regardless of whether the honest answer takes power away from your department, shrinks the budget you are responsible for, hands a capability to a team that is not yours, or — followed far enough — removes the reason your own role exists.

I have made a number of those calls, and they are not the enjoyable part of the work. Recommending the thing that made my own function smaller. Arguing against the direction my department had already committed to and half-announced. Telling a leadership team that the platform we had spent a year on was the wrong place to keep going. Moving against the stream in a room where everyone had good reasons to keep swimming with it, several of them mine.

That is the job, though. Anyone who only recommends what is comfortable for the part of the organization they happen to sit in is doing advocacy, whatever the title on the recommendation says. The whole reason the competence is worth having is that someone has to hold the entire picture, including the parts of it that are inconvenient for them personally.

Everything after that one is technical, and matters considerably less. Which languages we write in. Which cloud we run on. How we use AI, and where we have agreed not to. How a decision gets recorded. Individually none of these are interesting. What makes them guardrails is that they were settled deliberately rather than by whoever happened to start first.

The test for what belongs on that list is narrower than the one most organizations apply: standardize where variation costs more than it buys, and leave it open everywhere else. A second language is not a problem in itself. A second language nobody else can maintain, chosen by one team in an afternoon, is a hiring constraint and an on-call problem two years out. A second cloud is rarely worth what it costs. A second way of writing decisions down is almost never worth it, and costs nearly nothing to prevent.

The failure mode is a preference wearing the costume of a standard. Most arguments I have had about guardrails were not really about the guardrail — they were about someone's taste, defended in the language of consistency, because consistency is the one thing that sounds unarguable. The question that settles it is what it actually costs when teams choose differently. If the honest answer is "not much," it was never a guardrail.

That last technical one — how a decision gets recorded — deserves more than a line, because it has quietly become the only architecture documentation I still care about.

Not diagrams, which go stale the week after they are drawn. Architecture decision records: what we decided, why, what we considered instead, and what it costs us — written down at the time by whoever decided it, in markdown files sitting in the repository next to the code. A decision record nobody can find is the same as no decision record.

Three things come out of that, and none of them are documentation for its own sake. Decisions stop being re-opened, because the reasoning is available to anyone who wants to challenge it, and most people, having read it, don't. New people onboard against the reasoning rather than against the current state, which is a far quicker route to being useful than inferring intent from whatever happens to exist. And you can watch the architecture grow — read them in order

and you have the actual history of how the thing came to be this shape, rather than an archaeology exercise conducted by whoever is left.

The usual objection is that writing them is a chore, and I think that mistakes which part is the work. Making the decision is the job. Writing it down afterwards takes ten minutes.

Those ten minutes buy months. The alternative is passing every decision along mouth to mouth, and transmission like that drifts — each person keeps a slightly different version, fills the gaps with their own assumptions, and passes on the composite in perfectly good faith. Nobody is careless. The discrepancies simply accumulate, unnoticed, until two teams find they have spent a quarter building against different understandings of something everyone was certain had been settled.

The reason to keep the list short is that every item on it is a decision taken away from the people closest to the work. That is worth doing for the few things where it genuinely pays, and corrosive everywhere else.

And none of it holds if the guardrails were handed down rather than built together. Principles a team didn't help write are someone else's opinion wearing a badge of authority. Principles a team co-created are the team's own commitments, restated — and that difference shows up immediately in how the team treats them: as a rulebook to work around, or as the thing that makes their own decisions easier.

Guardrails also aren't a poster on a wall. They have to be visible in the actual rhythm of the work — onboarding, planning, retrospectives — and modeled first by whoever's asking the team to follow them. Teams follow behavior, not documents, and no set of principles survives a leader who doesn't act like it applies to them too.

A guardrail that makes the organization harder to change has failed at its job, however tidy it looks. Which means the list has to live where the work happens rather than in a gover-

nance portal, and has to be revisited — because the reason something was decided in 2019 is often not a reason at all by now.

What changes. The default flips. Instead of deciding what needs approval, you decide the few things that are settled — and everything else becomes someone's to decide without asking.

Write the list, and keep it embarrassingly short. If it does not fit on a page, it is not a set of guardrails, it is a policy manual with a friendlier name.

For each item, record what it would take to change it. A guardrail nobody can move is a rule. A guardrail with a stated route out is a decision the organization can revisit when the world does.

Retire one thing a quarter. Standards accumulate silently and are never audited for whether their reason still holds. Take the oldest item on the list and make someone defend it.

Put it where the work happens. A guardrail in a governance portal is a guardrail nobody reads. In the repository, the template, the checklist people already open — that one gets followed without anyone being asked to.

The half that points outward

All of that is friction — decisions that stop having to travel. There is a second effect, slower and easier to miss, and it is the one that eventually convinced me this was worth the trouble.

A team that knows what is settled has attention left over. Not time exactly — attention. The energy that was going into working out whether something needed asking, who to ask, and how to phrase it goes somewhere else, and where it

tends to go is the work itself. People start noticing things about what they are building that nobody had asked them to notice.

That is not a claim I can measure, and I would treat anyone who offered you a number for it with suspicion. But it shows up the same way every time: a few months after the list gets short, someone in the team raises a question about whether the thing they are building is right, rather than whether they are allowed to build it. That question was always available. It was simply competing for room with the other one.

Which is the case for keeping the list short beyond the obvious speed argument. Every item on it costs a small amount of somebody's attention, permanently, and attention is what value creation is made of.

Settle the few things worth settling. Leave the rest to the people doing the work, and write down why, so nobody has to decide it twice.

This cultivates: A Meeting Grew Here, in Part II.

EDITOR'S NOTES

The decision records described here are Architecture Decision Records, and the format comes from Michael Nygard's 2011 post *Documenting Architecture Decisions* — one file per decision, written at the time, kept next to the code. It has barely needed changing since, and I have never had a reason to improve on it. What this chapter adds is only where it sits in the argument: an ADR is the cheapest guardrail available, because it settles a question without removing any authority from the people closest to the work.

CHAPTER 42 · INTERFACES

Opening the Front Door

Making a capability reachable by someone who does not know anybody, which is most people.

BOTH · WORKS ON HANDOFF LOSS

You Have to Know a Guy ended with Maja, who is excellent, and who is also the reason a capability that looked available was only ever available to people who knew her name.

This is the most concrete cultivation in the paper, and also the least glamorous. Most of it is documentation. Documentation never feels urgent, never gets celebrated, and never stops paying — which is a poor combination for getting it funded and a very good reason to do it anyway.

What changes

You start treating *how do I use this* as part of the capability rather than as documentation about the capability. It is a small shift in wording with a large consequence: an interface nobody can find stops being a communication problem and becomes an unfinished piece of work.

Watch a newcomer try

The fastest way to find every missing interface in an organization costs one afternoon. Give a new starter a real task that crosses a boundary, ask them not to message anyone directly, and watch where they get stuck.

Every place they have to go and ask a human is a missing interface. They are completely invisible to everyone else, because everyone else already knows, and knowledge of this kind is impossible to un-know once you have it. This is why the newcomer is the only reliable instrument available — and why their value as one expires in about six weeks.

It is also a genuinely enjoyable exercise, as long as nobody treats the results as a report card on the teams involved. Nobody built those gaps deliberately. They are just what happens when helpful people keep answering.

Name a front door, then let it be boring

One route in, per capability, that a stranger can find and that does not require knowing a person's name. If the honest answer to *how do I request this* is "you ask Maja", then Maja is the interface, and Maja will eventually take a holiday, get promoted, or leave.

The reason to make it boring is that interesting interfaces attract maintenance. What matters far more is that it is the same one next quarter.

Which gives the order to work in, and it is worth being blunt about it because organizations reliably start at the wrong end.

Automate it, if it needs no human judgement at all. Most of what sits in a queue is a decision somebody already made once, being made again by hand. If the answer is a rule, the rule can be code, and the request stops existing rather than getting faster.

Failing that, make it self-service. Something the asker can call, query or grant themselves inside agreed limits — with the meaning of the thing written down next to it, because access without meaning is how a third version of the truth gets built.

Failing that, run a proper queue, with a name on it. Queues are not a failure. A well-run queue with a visible owner and an honest response time is far better than an interface nobody can find, and some things genuinely need a person.

And only then, a person you have to know. Which is where most organizations actually are, and which is not a strategy so much as what is left when nobody worked up the list.

The reason to start at the top is that each step removes work permanently rather than making it more pleasant. A faster queue still consumes somebody's week every week. An automated answer consumes an afternoon, once.

Then run the one-second test from An Interface Nobody Can Find across the front doors you already have, and mark each one honestly. If it lands in somebody's list, it is a queue.

Expect that exercise to be mildly deflating. Most organizations discover they have rather more queues than they thought and rather less of what they had been calling self-service. That is a good afternoon's work: nothing has got worse, and you now know which waiting is real.

Then the useful question for each queue is whether the judgement in it is genuine. Some of them exist because a decision really does need a person. Others exist because nobody ever wrote down what the answer is, and a queue formed over the gap — the same shape as the meeting in A Meeting Grew Here, one layer down. The second kind can usually be converted; the first kind should be left alone and staffed properly.

Organizations tend to build the escalation path first, because escalation is what hurts and pain gets budget. Most escalation routes exist because the ordinary route was never specified.

Publish internally as though you were publishing externally

An Interface Nobody Can Find makes the observation: internal and external consumers have identical needs and only one of them can take their business elsewhere. The cultivation that follows is a single standard rather than two.

A catalogue where anything can be found. Not a portal project. Somewhere a stranger can discover that a capability exists at all, which is the step most organizations skip because everybody internal already knows.

Documentation written for someone who has never spoken to the team. This is the hard part, and the test is not whether it is accurate — it will be. The test is whether it makes sense to a person with none of the context the author cannot remember acquiring.

Worked examples, because an example is what people actually copy, and they will copy whatever they find whether or not you meant it as a pattern.

Somebody who treats the experience of using it as part of the thing, rather than as an afterthought once the capability works.

The word for that discipline is developer experience, and it is the wrong word, because it makes it sound like a courtesy extended to developers. It is a friction control, and it belongs in the same conversation as ownership and guardrails.

And if any part of your business depends on other organizations building on top of what you do — the API economy in its unglamorous, everyday sense — the same discipline has to extend outward, published and discoverable. A partner who has to ask how to use you will find somebody easier to use, and will not tell you why.

The half this does for value

Everything above is friction. Here is the other half, and it is the reason I would put this chapter above its apparent importance.

An interface is where somebody outside gets to use what you can do. If it is hard to find, the people who give up are invisible: a partner who could have built on you and went elsewhere, an internal team that wrote their own version rather than wait, a customer who tried once. None of them file a complaint. They just do not come back, and nobody in the organization ever learns it happened.

So a findable front door is not only a handover you removed. It is the difference between a capability the organization has and a capability anyone can actually get value from — and those two are much further apart than most organizations believe.

Anyone building on you will tell you exactly this, if you ask them directly and make it comfortable to answer. They will not volunteer it.

Where it drifts back

Interfaces decay quietly. The rota stops being maintained, someone helpful starts answering directly because it is faster, and within a year the front door is decorative and Maja is back — usually a different Maja, doing the same generous thing.

That is not a failure of discipline. It is what happens when being helpful is easier than being findable, which it always is. Which is why this one is cultivation rather than a fix: it needs somebody looking at it about twice a year, forever.

This cultivates: You Have to Know a Guy, An Interface Nobody Can Find and The Detective Work at the Front of Everything, in Part II.

Different Capabilities, Different Conditions

The strictest requirement in the organization has a way of becoming everybody's requirement.

BOTH · WORKS ON OWNERSHIP CLARITY

Everyone Carries the Strictest Requirement describes what happens when a constraint written for one part of the organization becomes the constraint for all of it. This is the chapter about telling those needs apart on purpose.

It starts from something the observation only implies: different capabilities are *supposed* to run under different conditions. A capability handling highly sensitive information may need strong central control, a narrow set of approved technologies and considerable operational oversight. Another may need almost none of that. Both can be entirely right at the same time, in the same building — and an organization that cannot hold both is going to end up choosing one for everybody.

Boundaries are what make different choices possible

This is the less obvious argument for capability boundaries, and the one I now think matters most. They are not only a way of organizing responsibility — they are what lets an organization hold two different answers at once.

But that only works if the organization can actually separate them. If capabilities remain tightly coupled through shared data, shared infrastructure, shared processes and shared de-

cision structures, the strictest requirement will keep spreading across the boundaries regardless of what the diagram says. You will have drawn separate boxes without creating separation.

None of which is an argument for every capability becoming an island.

Quite the opposite, and this is the part that gets lost when people take the argument too far. Shared capabilities, shared infrastructure and shared standards remove enormous amounts of friction. Identity, security mechanisms, platforms, information standards, common services — most of these become both easier and safer when they are shared, and an organization that separates everything has simply chosen a different and more expensive kind of friction.

The question is never sharing versus separating. It is where sharing helps, and where it entangles needs that were never alike.

Share what is genuinely common. Separate where the requirements meaningfully differ.

And this goes well beyond technology. Different capabilities may need different degrees of autonomy, governance, resilience, security and oversight — and the goal is not to maximize any of them. It is to give each capability the conditions it needs in order to create value.

That requires boundaries strong enough to preserve real differences, while keeping the organization connected enough to work as one thing. The balance will never be right, exactly, and it moves. But without the ability to tell different needs apart, an organization ends up designing everything for its most demanding case, and everybody carries the friction.

The shape of the choice

Which is the general form of something this paper keeps arriving at from different directions, and it is worth stating plainly once.

The goal is not maximum autonomy, and it is not maximum control. It is the minimum structure needed to preserve coherence, while letting value be created with as little friction as possible.

Minimum is doing real work in that sentence. Not *no* structure — an organization with none is not free, it is merely uncoordinated, and it will rediscover why the constraints existed somewhere around month eight. And not the safest possible structure either, because safety applied uniformly is how the most cautious corner of the organization ends up running all of it.

The useful question for any constraint is therefore narrower than *is this a good control*. It is: which capability does this genuinely belong to, and what is it costing everywhere it has spread beyond that?

This cultivates: Everyone Carries the Strictest Requirement, in Part II.

CHAPTER 44 · TECHNOLOGY

Multiplying What's There

Technology is a poor first move and an excellent second one, which makes almost all of this about sequence.

BOTH · WORKS ON ALL FOUR

Faster, and Still Wrong made the observation: technology multiplies a capability and has no opinion about whether the capability was any good. This is the chapter about using that on purpose.

Which is mostly a question of sequence, and sequence is a much better piece of news than it sounds. Nobody has to give up on the platform. The argument is only about what happens in the eight weeks before it arrives.

What changes

Technology stops being the thing you buy to fix an organizational problem, and becomes the thing that multiplies whatever the organization already is. The buying decision gets easier once that is settled, because you are no longer asking the tool to do something tools cannot do.

Ask what it assumes before asking what it does

Every platform encodes a way of working. Who approves. What constitutes a team. Where a decision sits, and how many people have to touch a thing before it counts as done.

Those assumptions were reasonable in the organization the product was designed for, and they arrive in the box whether or not they match yours.

This is worth doing as an actual exercise rather than as a caution. Take the shortlist and, for each one, write down what it believes about how work gets approved. It takes an hour, it is more interesting than the feature comparison, and it occasionally eliminates a front-runner outright — usually the one that assumes a central function you spent three years dismantling.

The assumption also outlasts the contract. Long after anyone remembers choosing the tool, the organization will still be shaped a little like it.

Fix the ownership first, because the multiplication is honest

Scaling a capability nobody owns produces more of a thing nobody owns. This is the most reliable way I know to spend a large budget and change nothing, and it is very difficult to see from inside because every individual step is competent.

The good version of this is quick. Before scaling something, establish who would be able to say the output was wrong, and how long that answer takes to arrive. A name that answers in a day scales well. A committee that meets monthly is still an owner, but scaling the work without scaling the answering makes the forum the constraint on everything downstream of it. And a shrug scales into a much larger shrug.

Keep the map alive

Naming capabilities once, in a workshop, with cards, is the enjoyable part. The map is only worth something if somebody keeps it current as the organization changes shape underneath it — and it will, roughly every eighteen months, whether or not anyone updates the diagram.

A map nobody has touched in two years is a historical document. It is still interesting; it is no longer something to decide from.

Measure the path, and then the other end of it

Lead time for changes and deployment frequency are friction measures wearing engineering clothes. They are well defined, already collectable in most places, and the research linking them to commercial performance is stronger than anything else in this paper. If delivery is slow and ownership is already clear, *Accelerate* has far more to offer than I do.

But they only measure the pipe. An organization can get both numbers into the top decile and still ship, weekly and reliably, things nobody asked for. So the pairing worth building is one movement measure and one outcome measure, side by side on the same page: how quickly change reaches production, and whether anything changed for anyone once it got there.

The second number will be worse, later, and softer than the first. Keep them adjacent anyway. Two numbers of unequal quality, looked at together, produce better conversations than one excellent number looked at alone.

When it isn't the pipeline

If delivery is technically excellent and things still take nine months, the pipeline was never the constraint. Look at what had to be decided, by whom, and how far it travelled — and expect to find the answer somewhere in the first half of this paper rather than in any tool.

That is not a disappointing result. It is a considerably cheaper one than the platform would have been.

This cultivates: Faster, and Still Wrong, in Part II.

CHAPTER 45 · INFRASTRUCTURE

When Shared Infrastructure Becomes Shared Friction

Cloud, sovereignty and the exit nobody plans for — three versions of the same question about how much freedom to change is worth keeping.

BOTH · WORKS ON ALL FOUR

Different Capabilities, Different Conditions makes the case for telling needs apart. This is that argument where it is hardest to act on and easiest to measure: infrastructure.

For years the cloud conversation was framed as a destination. Should we move to the cloud? How much? Should we be cloud-first? For regulated organizations the answer was rarely simple, and the concerns were legitimate ones: where information is stored, who can reach it, which jurisdiction applies, how a service is recovered, what happens when a supplier becomes unavailable.

The problem was never that those constraints exist. Many of them should.

From cloud-first to the right cloud

A capability perspective changes the question being asked. Instead of *what is our cloud strategy*, it becomes: what does this capability actually require, and which infrastructure supports that?

Different capabilities need different levels of security, availability, resilience, data residency, operational control and technological independence. Which gives a more useful prin-

ciple than any strategy statement I have seen: **establish the level of sovereignty a capability genuinely requires, then choose the architecture that supports it.**

This thinking is becoming visible in European policy, which I find quietly encouraging. The Commission's proposed Cloud and AI Development Act frames different levels of cloud and AI sovereignty against risk and need, rather than assuming every workload requires the maximum. That distinction is the same one this chapter is making, arrived at from a completely different direction.

So a highly sensitive capability may need infrastructure under substantial European or national control. Another may be perfectly well served by a global provider. A third may be best served by a SaaS product where what you gain from the platform plainly outweighs the dependency it creates. There does not have to be one answer for the whole organization, and an organization that insists on one has usually chosen the most restrictive.

There is a catch, and it is the one the previous chapter named.

Different infrastructure choices are only available when the architecture lets capabilities be separated in the first place. Draw the distinction on a slide while everything still runs through the same database and the same deployment process, and you have described a separation you do not have.

The friction we hope never to feel

There is a side of this that gets very little attention: getting out.

Choosing a platform, a SaaS product or a provider concentrates almost all the thinking on getting in. Does it meet our needs? Is it secure? How fast can we start? What does it cost?

Which is reasonable — nobody picks a platform expecting to leave it, and with luck you never do.

But architecture has to hold the other question too. A change in regulation, in ownership, in pricing, in geopolitics or in strategy can turn a perfectly sensible choice into something that now has to change.

The friction built out of proprietary interfaces, data formats, licensing, tightly coupled services and simply the volume of data accumulated over years stays almost completely invisible until that moment. Then it is extremely visible.

European regulation is interesting here as a signal rather than as instruction. The Data Act introduced requirements meant to make switching between data-processing services easier. But the underlying architectural question is much older than any of it, and it is a good question to ask about anything:

How difficult have we made the next change?

Which is why exitability, in this paper's terms, is not really about planning to leave. It is about preserving the ability to change.

Dependency is not the enemy

There is an easy trap waiting at the end of that argument. If dependency creates future friction, perhaps architecture should minimize dependency.

It cannot, and it should not try.

Every modern organization depends on platforms, suppliers, cloud providers, open-source projects and each other, and those dependencies are frequently the entire reason it can move at all. Staying genuinely portable across every

provider creates a great deal of complexity today in order to insure against an event that may never happen. That is friction too — paid immediately, in exchange for a possibility.

So the question is not whether dependency exists. It is whether anybody understands the one they have.

For each capability there are two related questions, and they are worth asking out loud at the point of choosing rather than afterwards:

How much control do we actually need? And **how much freedom to change do we need to keep?**

The answers differ by capability, which is the whole argument of the previous chapter arriving in a different form. For one, accepting deep dependency on a provider is entirely rational, because the capability gains far more from that platform than it loses in optionality. For another, being unable to move would be unacceptable, and the extra complexity is the premium on an insurance policy somebody has consciously bought.

What architecture owes the organization is not the answer. It is making the trade visible at the moment it is being made, rather than eight years later when somebody discovers what was decided by nobody.

Infrastructure follows the need

Which lands back on the rule from the previous chapter, applied one layer down.

Shared infrastructure removes enormous amounts of friction. A good platform lets teams inherit security, identity, observability, deployment and operations without rebuilding any of it, and the organizations that have one are visibly faster than those that do not. Sharing stops helping at exactly the point where it forces fundamentally different needs into the same constraints.

The goal is neither maximum standardization nor maximum independence. It is infrastructure that gives each capability the conditions it needs, while keeping enough freedom for the organization to change — including the changes nobody has thought of yet.

Exit is friction we hope never to experience. The work is making sure it does not become friction we cannot overcome.

This cultivates: Everyone Carries the Strictest Requirement, in Part II.

EDITOR'S NOTES

The regulatory examples are current at the time of writing and will date faster than the rest of the paper. The EU Data Act sets requirements intended to make switching between data-processing services easier and to reduce obstacles to portability and interoperability. The European Commission's proposed Cloud and AI Development Act frames levels of cloud and AI sovereignty against risk and need rather than assuming maximum sovereignty everywhere. I cite them as evidence that the architectural question is being asked at policy level, not as advice on compliance — anyone acting on either should read the sources and take proper counsel rather than take my summary of them.

CHAPTER 46 · ARTIFICIAL INTELLIGENCE

Owning What AI Can't

Execution can be handed to a system. The part that can't is the part worth being deliberate about now.

BOTH · WORKS ON ALL FOUR

What AI Lands In described the same pilot in three organizations with three different outcomes, none of which the technology explained. This is the chapter about being on the good end of that, and almost all of it turns out to be about ownership — the same argument as the rest of the paper, made more urgent rather than different by how quickly software can now be produced.

What changes

AI stops being a procurement question and becomes an architectural one. The decision is not which tools to buy. It is whether the organization has anything coherent for them to amplify, and that question is answerable before anyone signs anything.

Check the ownership before the capability

Take three things AI would plausibly accelerate here. For each, ask who owns the outcome — not who runs the process, who is answerable for whether the result was any good.

Where that returns a name, acceleration is a straightforwardly good idea.

Where it returns a committee, the situation is more interesting than it first looks. A committee is an owner — it is simply a slower one, and one where the accountability is shared out until it is nobody's in particular. That can be entirely correct: some decisions genuinely should be made by several people who each see a different part of it, and the slowness is the cost of getting it right.

What it cannot do is answer at the speed the acceleration will demand of it. Make the work ten times faster and a monthly forum becomes the whole system's clock. So the question is not whether the committee counts as an owner. It is whether it can answer as often as it will now be asked, and whether anybody in it will stand behind the answer afterwards.

None of which is an argument for waiting. It is an argument for spending two weeks on that question first, which is cheaper than the pilot and considerably cheaper than the eighteen-month conversation.

Assume the building gets distributed

More people are going to create more software, in more corners of the organization, sooner than most plans assume. Some of it will be very good. Some of it will be a spreadsheet's worth of business logic living on somebody's laptop, which is not new — it is just faster now.

The useful question is not whether to allow it. It is which guardrails and interfaces make it safe enough to be a good thing, and those take longer to establish than the tools take to arrive. That gap is the whole planning problem, and it is the one part of this worth being early on.

What the technology function is for once translation is cheap

For a long time a large part of what technology functions did was translate: turning what the business wanted into something a system could do. That was scarce, valuable, and is becoming much less scarce.

This is not a defensive question, and the good answers are more interesting than the queue ever was — stewardship of the parts that are genuinely hard, guardrails, interfaces, and the capabilities where being wrong is expensive. They are worth deciding deliberately rather than discovering by attrition when the queue empties on its own.

Keep a human on every outcome

Execution can be delegated to a system. Ownership cannot. The moment it is assumed rather than assigned, it is gone, and the failure mode is specific: everybody believed somebody else was checking.

This is the one line in the chapter I would defend without qualification, and it applies at every scale — a generated report, a decision support tool, an agent doing something on its own. Not because the technology cannot be trusted, but because accountability is a relationship between people, and there is nobody on the other end of it to be in that relationship with.

The half that points outward

Everything above is about not making things worse quickly. Here is the other half, and it is the more hopeful one.

Speed of production is now, for the first time in my career, not the scarce thing. What remains scarce is knowing what is worth producing, and finding out whether it landed — both of which are human, both of which are slow, and neither of which gets faster because the building got faster.

Which means the organizations that will get the most out of any of this are the ones that already have a short distance between the people doing the work and the people affected by it. That distance was always worth shortening. It is now the constraint.

As technical friction keeps falling away, what is left is the organizational kind. That is the argument the whole paper rests on, and AI is simply the clearest demonstration of it any of us have been handed.

This cultivates: What AI Lands In, in Part II.

CHAPTER 47 · FUNDING

Funding What Doesn't End

The project is a funding decision with a dissolution date attached, and the dissolution date is where most of the trouble lives.

BOTH · WORKS ON OWNERSHIP CLARITY

The project closed on a Friday. There was a slide with a green square on it, a short round of thanks, and a cake that somebody had clearly bought on the way in. By the following Wednesday the people who built the thing were spread across four other initiatives.

About seven months later a question came back. Had it worked — the thing we spent a year and a good deal of money on, had it actually changed anything for the people it was built for?

Nobody could say. Not because anyone was hiding, and not because the answer was bad. There was simply no longer anyone whose job it was to know. The team that would have noticed had been dissolved on schedule, which is what teams are supposed to do when a project ends. Everyone had followed the process exactly.

I have watched that closing Friday a lot of times, and what is worth noticing is that the process was working perfectly and producing the outcome anyway.

What a project actually bundles

We use the word project for something quite specific: a scope, a budget, a group of people, and an end date, tied together and approved as one decision. That bundle is useful. It

makes work legible to finance, it gives someone something to report on, and it lets an organization say yes to one thing rather than to everything.

It also decides, at the moment of approval, that this will stop. And once you look at the bundle as a piece of structure rather than as an administrative wrapper, it is doing a great deal more than accounting.

A project draws a boundary around itself, because that is what a scope is. Everything outside the boundary now has to be reached across, which is the definition of a handover. It stands up a steering group, because a boundary needs somewhere for decisions that cross it to go, and that forum meets monthly, which sets the pace of every question that reaches it. It appoints an owner whose ownership expires. And it names a moment called done, which occurs before anyone outside the organization has had a chance to react to what was built.

That is not an unfortunate side effect of projects. That is what a project is. The Introduction says a transformation programme manufactures friction while it runs; this is the same observation at a smaller scale, and it applies to a €200,000 initiative as neatly as it applies to a three-year programme.

The part that is harder to see

The handovers and the steering group are visible enough that people complain about them. The dissolution date is the expensive one, and almost nobody complains about it, because it looks like tidiness.

Value arrives late. That is not a failure of anyone's planning — it is a property of the thing. People have to adopt something, get used to it, work out what it is actually good for, and change what they do. Six months is quick. A year is normal.

Projects end before that. So the moment where the answer becomes available is, reliably, several months after the last person who cared about the question was reassigned. The organization does not decide to skip learning. It funds a structure that makes learning arrive at an empty room.

And because nothing contradicted anyone, the next initiative gets chosen the same way, by the same people, with the same confidence.

The other unit

The alternative is not complicated, which is part of why it is hard. You fund a team, for a period, against an outcome, and you let the work come to the team instead of assembling a team around the work.

The team persists. It carries the context rather than documenting it, so the handovers that a project needs simply are not there to be paid for. It is still around when the outcome shows up, which means somebody can be asked and can answer. And because it has no end date to sprint toward, the question of whether the current work is still worth doing is one it is allowed to raise, which is the question a project is structurally incapable of asking about itself.

This is well-trodden ground outside this paper — it is roughly what product organizations mean by a product team, and roughly what *Team Topologies* means by a long-lived stream-aligned team. I have nothing to add to either description. What I would add is that most of the benefit shows up as friction that stopped happening, which makes it very difficult to point at in a business case.

If you want to see the size of it in your own organization, pick something delivered eighteen months ago and try to find the person who can tell you what it changed. Not what was delivered — what changed. How long that takes, and how many people you have to go through, is the measurement.

Where this gets applied badly

Some work genuinely ends. A data centre migration ends. A regulatory deadline ends. Merging two payroll systems ends, thank God. Running those as projects is correct, and an organization that has decided everything is a product will eventually make a mess of the things that were finite all along.

The test is not whether the work has an end. It is whether the *outcome* has one. Nobody keeps asking whether the payroll merge is still delivering value; the value was a working payroll system and it either works or it doesn't. But a customer portal, a claims process, a service anybody uses more than once — those have outcomes that keep going, and if the team ends while the outcome carries on, the outcome is now nobody's.

The other failure is renaming. An organization keeps annual funding rounds, keeps the scope-shaped approval, keeps the completion report, and calls the result a product team. Nothing has changed except the noun. The team still has a fixed scope, still has to go to a forum for anything outside it, and still gets rearranged at the end of the year. Whether the change is real is decided in the funding model, not in the org chart, and the funding model is owned by people who mostly do not read papers about architecture.

What it costs to do properly

A durable team has no end date to point at, which makes it look, on a spreadsheet, like a permanent cost with no defined return. A project looks like a bounded one. That comparison is wrong in a way that is genuinely difficult to argue against in a budget meeting, because the project's boundedness is exactly the thing producing the loss, and the loss is invisible.

I have not found a clever way around this. The version that has worked is unglamorous: keep one team standing for two funding periods, be strict about asking what changed at the end of each, and let the answers be the argument. The first period produces very little worth showing. The second is usually where somebody senior notices that this is the only part of the organization that can answer the question.

This cultivates: What Survives the Reorg, in Part II.

EDITOR'S NOTES

The argument that the project is the wrong unit, and that the funding model produces the behaviour rather than the people, is Melissa Perri's in *Escaping the Build Trap*. She makes it considerably more thoroughly than this chapter does, including the part this paper skips — what an organization has to change above the team, in strategy and in how work is chosen, before any of it holds. Long-lived teams as the durable thing, with work flowing to them rather than teams assembling around work, is Skelton and Pais's in *Team Topologies*. Anyone who wants the mechanics should read both.

What this chapter adds is the friction reading. Perri's case against projects is mostly about learning and about strategy. The case here is structural: a project assembles the four shapes on purpose — a boundary that becomes a handover, a steering group that becomes decision latency, and a dissolution date that removes the owner before the outcome arrives. That reading is what connects the funding model to the rest of this paper rather than leaving it as a product management argument borrowed into an architecture one.

CHAPTER 48 · LEARNING

Closing the Loop

The shortest, cheapest stage in the whole circulation, and the one almost everybody skips.

BOTH · WORKS ON LEARNING CYCLE TIME

We Have Solved This Before names three reasons the loop stays open: the outcome arrives somewhere other than the decision, the learning is owned by a document rather than a person, and admitting the lesson costs somebody something. This chapter is about those three, in that order, because they are not equally hard and organizations reliably attack them in the wrong sequence.

The wrong sequence is to start with the third. Announce psychological safety, run a blameless post-mortem, and hope the honesty arrives. It occasionally works and it usually produces a room where everyone is being very careful about being blameless.

The first two are structural, and doing them first makes the third considerably cheaper.

Send the outcome back to whoever chose

The most common reason nothing is learned is that the people who found out are not the people who decided.

The team who built it know exactly how it landed — they hear from users, they see the support tickets, they watch the adoption curve flatten. The people who chose to fund it have moved on to next year's portfolio and will encounter this decision again only as a line in a review deck.

So the practice is unglamorous: when an outcome arrives, route it to whoever made the call, by name, and do it whether or not the news is good.

That is it. No forum, no process. The thing that makes it work is the *by name* part, because a report addressed to a distribution list is addressed to nobody, and everyone on it assumes somebody else is the intended reader.

I have watched this change behaviour faster than anything else in this part of the paper, and the mechanism is not shame. It is simply that people who find out how their decisions turned out start making different decisions, and people who never find out cannot.

Give the lesson an owner, not a document

A retrospective produces actions, the actions go into a list, and the list has no owner — which is how a genuine insight becomes an unread row in a spreadsheet within a fortnight.

The fix is to stop producing lists. One lesson, one name, one date. If nobody will take it, it was not a lesson; it was a complaint, and it is better to say so in the room than to bury it in a document where it will look like progress for six months.

Which means most retrospectives should produce fewer outputs than they currently do. A session that generates eleven actions has generated none. A session that generates one thing somebody will actually do has changed the next quarter.

And write down what you expected, not only what you decided. This is the same discipline that Measuring Value asks for, one layer down. A decision record with no expectation in it cannot be checked later, because there is nothing to check it against. *We think this will do X by March* costs one sentence and converts a future argument about interpretation into a factual question.

Then make the third one survivable

Once outcomes reach deciders and lessons have owners, you will eventually hit the case where the honest conclusion is that a senior person got something wrong.

This is where psychological safety stops being a pleasant idea and becomes an operating requirement — and not in the comfortable sense. It is the specific ability to say what happened without it costing the person saying it, and it is produced almost entirely by what visibly happens the first two or three times somebody does.

Which puts most of the weight on one thing: whether the senior person can say it about themselves first. Not as a performance of humility at an all-hands. In the ordinary meeting, about the ordinary decision, in the same tone they would use about anyone else's.

I have seen that done well, and the effect is immediate and slightly startling — the room recalibrates in about ten seconds. I have also failed to do it, more than once, and the cost is not that people call you out. It is that they quietly stop bringing you the difficult half of what they know, and you do not find out for a year.

Let the loop close on things that went well, too

Organizations examine failures and celebrate successes, which means half the available learning is being applauded rather than examined.

The question after something worked is the same question: what would we do again on purpose, and what happened by luck? Teams are usually honest about this if asked, and almost never asked, because the celebration is the point and nobody wants to interrogate a good quarter.

What it costs

Very little, which is the strange part, and I think it is why the loop stays open. Nothing here needs budget, headcount or a programme. It needs somebody to send a message they were not going to send, and somebody senior to say a sentence that is mildly uncomfortable.

The reason it does not happen is not cost. It is that no single instance seems worth the effort — and the return arrives a year later, in decisions that were never made, which nobody will ever attribute to this.

An organization that closes the loop learns faster, and the part you notice first is that it stops paying for the same lesson twice.

This cultivates: We Have Solved This Before and It Breaks in the Joints, in Part II.

Reducing Friction

Removing friction before adding capability.

REDUCES FRICTION · WORKS ON ALL FOUR

One principle, and it's a subtraction: **remove friction before adding capability.**

The instinct in most organizations facing slowness is to add — more people, more tooling, another initiative, another coordinating role. Adding is visible, fundable, and reportable. Removing is none of those things, which is why it's rarely anyone's proposal even when it's obviously correct.

But removing friction is the only way to go faster that costs nothing and compounds, because you aren't adding energy to the system, you're stopping the system consuming it.

Practically, that sequences into three moves. The order matters more than the moves do.

First: make ownership unmistakable

Almost everything else is downstream of this. Committees created to compensate for unclear ownership become removable the moment ownership is clear — a genuine subtraction with no replacement needed, which is rare enough in organizational change to be worth going after first.

How it goes wrong. Ownership gets "clarified" by writing it down. Someone produces a responsibility matrix, circulates it, and nothing changes — because a document does not create the condition it describes. Ownership is not a claim about

who is accountable. It is a state in which the named person can decide without asking, knows it, and everyone around them knows it too.

The test is not whether a name exists. It is whether that person has ever overruled someone more senior on the thing they supposedly own, and had it stand. If not, the ownership is decorative.

What to do instead. Name the owner out loud, in the room, in front of the people it affects. Then take a decision that used to escalate and let it not escalate. The second half is the entire exercise; the first half is a memo.

Second: move the decision to the information

Not the information to the decision.

Most organizations try it the other way, and it is worth being precise about why that fails. Improving reporting so a distant decider is better informed adds a step to a chain that was already too long. It also cannot work in principle: the summary that reaches the decider is always smaller than what the person on the ground knew, and the difference between them is exactly the judgement you were hoping to buy.

How it goes wrong. Authority is delegated in a memo and retracted in practice. The first time a delegated decision produces a bad outcome it quietly returns upward, and everyone learns that the delegation was conditional on being right. One retraction costs more than ten delegations build.

What to do instead. Delegate the decision together with an explicit tolerance for it going wrong, stated in advance and in public: *this is yours, some of these will be wrong, and a wrong one does not send it back to me.* Then survive the first bad one without taking it back. There is no other way to do this part.

Third, and only then: scale with technology

The question from Technology applies: what capability does this multiply, and is that capability currently good enough to be worth multiplying?

How it goes wrong. This step gets done first, because it is the only one of the three that can be procured. A tool is a decision you can make in a quarter with a budget; the other two require confronting people about authority. So the tool arrives, the friction stays, and the organization concludes — reasonably, from the evidence in front of it — that the tool didn't work.

When the boundary will not move

Some boundaries are fixed. A regulator requires them, a joint venture created them, an acquisition never finished integrating, or someone's position depends on the boundary staying exactly where it is. Pretending otherwise costs a year.

When a boundary can't be removed, the goal changes from removing the handover to making it cheap:

- **Make the interface explicit.** If work must cross, publish how — documented to the standard you'd use for an outside consumer.
- **Move the wait off the critical path.** A handover that happens in parallel costs elapsed time once, rather than every time.
- **Reduce the frequency rather than the cost.** Ten cheap crossings can be worse than two expensive ones.
- **Name it as a known cost.** Put a number on it and revisit it annually. Boundaries nobody questions outlive their reason by a decade.

What improvement feels like from the inside

Nothing dramatic, and people should be warned about that in advance.

An organization that has removed real friction does not feel fast. It feels ordinary. A meeting stops being scheduled and nobody remarks on it. A decision gets made in the room where the problem came up. Someone notices that a thing they used to dread has become unremarkable.

The absence of a wait doesn't announce itself, which makes improvement nearly impossible to celebrate and very easy to under-claim. That is precisely why the numbers matter. Without a before and an after, nobody will believe anything happened — including the people who did it.

Say it back before anyone leaves the room

Everyone Left Agreeing is the cheapest friction in this paper to remove and the most reliably skipped, because removing it costs a small amount of social nerve at exactly the moment nobody has any spare.

Say it back. Not a summary of the discussion — a statement of what you are going to go and do, in your own words, out loud, while everyone who could correct you is still present. *So I'm going to do X, and I'm assuming Y is out of scope.* Half the time everyone nods and you have lost ninety seconds. The other half, somebody says wait, Y is definitely in scope, and you have recovered three weeks.

Write down what was assumed, not only what was decided. Decisions get minuted almost everywhere. Assumptions almost never do, and assumptions are the part that diverges. A

decision record with no assumptions in it tells a future reader what was chosen and nothing about the world in which choosing it made sense.

Treat "we're aligned" as a hypothesis. It is an accurate description of how a conversation felt. It is not evidence that eight mental models match, and it is most likely to be said in exactly the meetings where they don't, because a discussion with no friction in it is a discussion where nobody surfaced their picture.

The reason this works better from the top of the room is worth stating plainly: whoever has the most standing pays the smallest social cost for going first. If you are the senior person present and you say it back before anyone else does, you have made it free for everyone else. If you never do, you have made it expensive, whatever you have said about psychological safety.

All of which recovers capacity, and recovers it permanently. It says nothing whatsoever about what the capacity should then be spent on, and an organization that gets this half right and stops here has built a very efficient way of arriving somewhere it never chose.

This cultivates: Everyone Left Agreeing, in Part II.

CHAPTER 50 · VALUE

Creating Value

The other half of the work, cultivated as deliberately as friction is removed.

CREATES VALUE · WORKS ON LEARNING CYCLE TIME

The best-run team I have worked with built something nobody wanted.

They were genuinely good. Ownership was unmistakable, the boundary was drawn around a real capability, and they had the business judgement embedded rather than represented — the arrangement the whole of Part II argues for. They had cut their own lead time by more than half in a year, without being asked to. Nothing about them was broken.

Eighteen months later most of what they had built was switched off. Not because it failed. Because the thing it did turned out not to matter much to anyone, and the organization had been so pleased with how fast it was arriving that nobody had asked.

That is the failure this chapter exists for, and it is the one the previous chapter cannot prevent. Reducing friction makes an organization produce more. It has nothing to say about what.

Both halves are cultivated

There is a tempting and wrong way to hold this: friction is the thing you work on, and value is the thing that arrives afterwards if you did the first part properly. Remove the obstacles, and the good outcomes take care of themselves.

They don't. An organization can be cultivated toward greatness on both halves, and the two require different attention.

Friction is cultivated by **subtraction** — a boundary settled, a decision moved closer to the information, an approval removed because ownership made it redundant. You are taking things out of the way of something that already wants to happen.

Value is cultivated by **contact** — with the person on the other end, with the outcome rather than the output, with evidence rather than the assumption written down at funding. You are not removing an obstacle. You are shortening a distance.

Neither substitutes for the other, and an organization that only ever does the first will get very fast at something, eventually at almost anything, including the wrong thing. The compliment I have heard most often about high-performing teams — *they just ship* — is a description of one half of a system, delivered as though it were the whole.

Why the second half gets postponed

Value is only legible from outside the organization. That single fact explains almost everything about why it goes unattended.

Output can be counted on a Thursday by someone in the building. Value requires asking someone who does not work for you, waiting for an answer that arrives late, and accepting a number that refuses to attach itself cleanly to any one team's effort. Given a choice between a measurement you can produce this week and one you can't, the week wins, in every organization I have worked in.

So the question of whether any of it produced anything gets answered once, at funding, in the one moment when no evidence exists yet — and then never again, because by the time evidence exists the decision is a year old and nobody wants to reopen it.

Four things that actually work

None of these are programmes. Each one is small enough to do without permission, which is deliberate.

Put someone in the room who experiences the outcome. Worth being precise about who that is, because it is easy to believe you have already done it.

Internal customers are real customers. When one capability hands to the next — underwriting to policy administration, recruitment to onboarding, claims to payments — the receiving capability genuinely consumes what the first one produced, and it deserves the same interface, the same documentation and the same attention you would give somebody outside who was paying. Nothing in this paper argues otherwise.

What it does argue against is a different arrangement that dresses itself in the same language: "the business" requests and IT delivers. That is not a customer relationship, because those two are not two capabilities. They are two halves of one, split into a requester who owns the outcome without the means and a supplier who owns the means without the outcome — which is why the vocabulary of service never quite fixes it. The Team That Has to Ask sets out what that split costs.

So there are two questions here rather than one, and an organization can be excellent at the first while completely blind on the second. Who consumes what we produce, and are we treating them as well as we would treat a stranger. And then: how many hops sit between us and somebody outside the organization altogether. In a platform or an ecosystem the people you talk to are usually the next team along, and they will ask you about fields, rate limits and edge cases. That conversation is valuable and it is not this one, because nobody in that room has met the person at the far end either.

With that said, this is the single highest-return move in the paper, and it appears twice for a reason — once as a friction fix in The Cost of Value Never Created, and again here. When the finance controller sat next to the team, the printouts went away; that was the friction half. The part I did not expect was that they started building things the controller asked for that nobody had specified, because the conversation was cheap enough to have. Proximity removes handovers and it generates value, and it is the same intervention both times.

The same thing worked running the other way, under a name I have kept: *guest of reality*. Developers spent time out where the work actually happens — beside somebody using what they had built, in the operation rather than in a workshop about the operation.

What came back was not the strategic insight anybody had hoped for. It was a pile of small things. A field re-entered three times. A report exported and reformatted by hand every Monday. A screen that made sense only if you already knew the answer. A good number were fixed within the week, because the person who could fix them had now seen them — and none of them would ever have been written down and submitted. They were each too small to be worth anybody's request form, and together they were most of somebody's week.

It also settled the estimating, in both directions, which I had not expected. Things the business had assumed were enormous took an afternoon. Things assumed trivial turned out to touch four systems and a contract nobody had read recently. Neither side could have known that on their own, and what resolved it was almost always somebody saying *wait, say that again — why does it have to work that way*.

Which is the part worth keeping. The fixes were good, and the conversation was the point. Cross-competence is not produced by putting the competences on the same org chart. It

comes from them having enough of those conversations to stop guessing at each other's half.

Fund outcomes for a period, not outputs on a date. Most organizations fund a scope and then measure whether the scope was delivered, which guarantees the answer is about delivery. Fund a team against an outcome for two quarters, and the conversation at the end is about whether anything changed for anyone. It is a harder conversation. It is also the only one that ever produced a decision to stop.

Make someone own the outcome, not the delivery. Ownership of a delivery ends when the thing ships. Ownership of an outcome does not end at all, which is precisely why organizations avoid assigning it. But an owner who is still holding the question six months later is the only mechanism I know of that reliably kills work that stopped being worth doing.

Close the loop, and let the closing be uncomfortable. Learning is the return leg. If the outcome never travels back to the people who chose the work, they will choose the same way again, confidently, because nothing contradicted them. The test is the one from the Learning chapter: after the last significant thing that did not land, what is now decided differently?

The trap, which is easy to fall into

Value work fails in a characteristic way, and it is worth naming because it looks like success.

An organization decides to focus on outcomes, discovers that outcomes are hard to measure, and adopts a proxy — engagement, adoption, usage, satisfaction, a score out of ten. The proxy is countable, so it gets counted, so it gets reported, so it gets targeted. Within a year the organization is optimizing

the proxy with the same efficiency it previously applied to output, and the distance to the actual person has not shortened by a metre.

The defence is not a better proxy. It is keeping at least one channel open that cannot be optimized: someone in the organization who talks to real users often enough that the numbers can be contradicted by a specific human being. That channel is cheap and it is the first thing cut when the quarter gets tight.

What it looks like when both halves are cultivated

Work arrives faster and less of it is wrong. Fewer things get built, and more of what does get built is still running two years later. The organization's arguments move upstream — from *why is this taking so long* to *is this the right thing* — which is a considerably more useful argument to be having, and one that only becomes available once the first question stops dominating the room.

And the two halves compound. Removing friction shortens the distance between deciding something and finding out whether it was right, which makes the learning loop tighter, which improves the next decision about what is worth doing at all. Cultivating value gives the recovered capacity somewhere worth going. Neither one on its own gets you there.

An organization that only removes friction becomes efficient. An organization that only chases value becomes well-intentioned and slow. The ones worth working in have been patient enough to grow both.

Which leaves the practical problem of finding out where you currently stand on either one, in a room full of people who have every reason to tell you it is fine.

This cultivates: Everything Was Green, in Part II.

PART IV

Applying Organizational Flow

Where would you begin, with
what is in front of you on Monday?

Where to look first, how to measure both halves, what I have got wrong doing this, and what it looks like to work this way over years rather than quarters.

Organizational Flow is a way of seeing rather than a template, which is the best thing about it. It stays a working theory, deliberately unfinished, and it gets better every time somebody argues with it.

CHAPTER 51

Observing Organizations

Four places friction becomes visible — as a number if you have the time, as a question if you don't.

BOTH · WORKS ON ALL FOUR

Flow is hard to look at directly and surprisingly easy to notice sideways, through the traces it leaves. The four shapes friction takes, named in The Cost of Value Never Created, are the same four worth watching for — and each one can be looked at two ways.

One is a number: deliberately narrow, collected the same way twice, useful for a trend line. The other is a question: askable in the room you're already in, with no instrumentation and no delay, useful the moment someone reaches for it. Neither replaces the other. The number is what you build a case with. The question is what you notice with, this week, before there's a case to build.

Decision latency

What you are actually looking at. Elapsed time from a decision becoming necessary to that decision being made, in calendar days — the weekend is part of the wait for whoever is blocked by it. The clock starts at the first recorded request and stops when the decision reaches the people who have to act on it, not when it was made in a room nobody outside it heard about.

How to get hold of it. Take the last ten cross-team decisions — ten rather than five, because the distribution matters more than the average. For each, record start, stop, and how much of the gap was deliberation rather than waiting. Report the

median and the worst case, never the mean. In a healthy organization deliberation and waiting are close. In most, deliberation is minutes and waiting is weeks, and the ratio between them is the actual finding.

Or just ask. *What did the last decision at this level cost in waiting, as opposed to in deliberation?* A bad answer sounds like "these things take time." Split the total, out loud, in the room — the ratio is usually a fact nobody there has seen before.

Handoff loss

What you are actually looking at. Effort spent rebuilding context that already existed, each time work crosses a boundary between groups, in person-hours per handover. Ask the receiving team, about one specific piece of work: how long before you could act without going back to ask? Count meetings, message threads, and anything the sending team had already settled that got settled again.

How to get hold of it. Multiply by frequency. Four hours of reconstruction on a boundary crossed twice a year is a rounding error. The same four hours on a boundary crossed weekly is most of a full-time role, spent entirely on remembering. This is the softest of the four numbers — it relies on self-report, and people systematically under-count work that feels like helpfulness. Treat it as an order of magnitude rather than a figure.

Or just ask. *Is this boundary here for a reason that still holds?* A bad answer sounds like a history lesson. Reorganizations, acquisitions and one departed executive's preference all leave boundaries behind, and those boundaries outlive their reason by an average of about a decade. If nobody can name a current reason, you have found something.

Ownership clarity

What you are actually looking at. Whether responsibility for a capability is unambiguous to the people around it. Name one capability. Ask five people in different parts of the organization, separately and without preamble, who owns it. Record the answers verbatim.

How to get hold of it. One answer, confirmed by the person named, is clear. Two answers is a boundary dispute. Three or more is where governance will grow next, whether or not anyone decides to allow it. A cheaper variant: ask a team whether they deliver *to* the business or *with* it, and listen for which preposition they reach for without thinking — ten seconds, and right more often than it has any business being.

Or just ask. *What must we always be able to do here — regardless of which system, team or vendor happens to be doing it now?* A bad answer sounds like a system name, a supplier, or a department. If the answer can't survive replacing the tool, it was a description of the current arrangement rather than of the capability.

Learning cycle time

What you are actually looking at. Elapsed time between an outcome and a decision that changed because of it — or, honestly, a binary: did any decision change at all. Take the last significant thing that went wrong. Name the specific decision that would now be made differently, and find the date it changed. Not the date of the retrospective. The date the changed decision took effect.

How to get hold of it. If no such decision exists, the number isn't large — it's undefined, and the loop is open. That is the most common result and the most expensive one, because it means the organization is paying full price for the same mistake more than once.

Or just ask. *What changed as a result of the last thing that went wrong?* A bad answer sounds like a document, an action list, or a process that was "tightened." Push for a named decision that would now go differently, and a date it took effect. If neither exists, the same failure is still fully funded.

What these are, and are not

The numbers are indicators, not targets. Managed as targets they get optimized directly and stop measuring anything, which is the usual fate of any organizational number that acquires a bonus. Decision latency in particular is trivially gamed by deciding faster and worse. Nor are they benchmarks — none of this has been measured across enough organizations for anyone to know what a good value would be. The number is only useful against your own earlier number, which means the first measurement is worth very little on its own, and the fourth is worth a great deal.

The questions carry a different risk: they stop working the moment they read as an audit. Three things keep them on the useful side of that line. Ask about one concrete instance rather than the general case — *how did the last one go* produces evidence, *how do we handle these* produces a description of the process. Ask people about their own experience of waiting, not about someone else's performance; nobody is defensive about having waited. And be willing to be the answer — the question about who is waiting on you is the most valuable one on this list, and it is only safe to ask if you are visibly prepared to hear your own name.

Four is deliberately few, and each is a summary of several things this paper has spent chapters on. Ownership clarity sits on top of purpose, guardrails, and whether anyone treated the last reorganization as an architectural decision. Handoff loss sits on top of team composition and whether a capability was split across two groups. Decision latency sits on top of information reach, funding rhythm, and whether

managers are still the place decisions go. Learning cycle time sits on top of everything, which is why it is usually the worst of the four.

That is the useful property of the four, measured or asked. They are not separate observations — they are one condition, visible from two places, which means you can start with either and end up in the same conversation.

EDITOR'S NOTES

The four numbers are diagnostic prompts, not validated metrics, and the difference matters. They come from noticing the same patterns across twenty-five years of work, not from a research programme — there is no dataset behind them, no construct validation, and no evidence that a particular reading predicts a particular outcome. Where measured research does exist it is worth going to directly: DORA's four keys (Forsgren, Humble and Kim, Accelerate) are properly operationalized and statistically validated for software delivery performance. This chapter sits deliberately upstream of those, measuring conditions rather than delivery, and nothing here carries the same evidential weight.

Decision latency is lead time applied to decisions rather than to work, and handoff loss is a restatement of flow efficiency. Both are borrowed from Lean and from the Theory of Constraints, where they have been defined and measured far more rigorously than they are here.

CHAPTER 52

Measuring Value

The fifth measure, and how to reach it from where most organizations actually stand.

CREATES VALUE · WORKS ON LEARNING CYCLE TIME

Every organization I have worked with could tell me what it delivered last quarter. In some cases to two decimal places, with a chart.

Almost none could tell me what any of it changed.

That is not a criticism, or not only. It is what happens when one number is available on a Thursday and the other one isn't available for a year, and something has to go in the report on Friday. Delivery gets called value because delivery is the number that exists.

The four indicators in the previous chapter all measure movement — who owns it, how often it changes hands, how long it waits, whether anything came back. An organization can improve all four, genuinely and measurably, and produce more of something nobody wanted, faster than before. That is why there is a fifth, and why it sits apart from the others: it is the only one that points outward.

The substitution, and how to spot it

The substitution is rarely a decision. It happens in the wording.

Somebody writes *delivered the new onboarding flow* under a heading that says Value. Somebody else reports *90% of the migration complete* as a benefit. A slide says *value delivered this quarter* and then lists things that were built. Nobody is

lying. Everyone would agree, if asked directly, that shipping a thing is not the same as the thing being useful. The heading just quietly does the work of the argument.

The reliable way to catch it is to read your own reporting and ask, of each line, whether it describes something that happened inside the building or outside it. Almost everything will be inside. That proportion is the honest starting position, and it is worth knowing before trying to change it, because it tends to be more lopsided than people expect.

Four rungs

What follows is not a maturity model, and I would rather it were not drawn as a pyramid. It is just the order I have seen organizations actually move in, and each rung is a real place to stand for a while.

We delivered it. Countable immediately, entirely under your control, and it tells you nothing about effect. Most reporting lives here. The only thing wrong with it is the label — called output, it is useful; called value, it stops anyone looking further.

Somebody used it. Adoption, usage, the thing being opened. This is a genuine step, because it is the first number that requires another person to do something. It is also where a lot of organizations settle permanently, since it is countable from your own systems and feels enough like evidence. It isn't: people use things they dislike, daily, because it is the only way to get paid.

Somebody outside says something is different. The first measure that leaves the building. It is late, it is fuzzy, it will not attribute cleanly to one team's effort, and it is the first one that could actually contradict a decision you made.

Something is decided differently because of it. The rung that matters. A measure that cannot change a decision is decoration, however rigorous it looks. If the answer arrives and the roadmap is identical either way, you have built a reporting obligation, not a feedback loop.

Most organizations are trying to jump from the first rung to the fourth, usually by way of a large dashboard, and stall. Moving one rung is enough for a year.

The smallest version that works

If there is one practice in this paper I would keep when everything else was cut, it is this one, and it costs nothing.

Before the work starts, write one sentence: *we think this will let [these people] [do this thing] better, and we expect to see it by [when].* Not a business case. One sentence, written where it can be found again, by the people choosing the work rather than by anyone in a planning function.

Then, at the date, go and look.

That is the whole practice. It is a smaller cousin of what Melissa Perri calls the product kata, and I recommend the original for anyone who wants it done properly. What makes even this stripped-down version work is not the measurement, which is usually rough. It is that somebody committed to an expectation in writing, in advance, which is the one condition under which being wrong is informative rather than embarrassing.

The organizations I have watched fail at this mostly failed at the second half. Writing the sentence is easy and mildly enjoyable. Going back seven months later, when everyone has moved on and the answer can only be awkward, is a habit somebody has to protect.

Asking, when you cannot count

Value often cannot be counted honestly. It can nearly always be asked about, and the asking is worth more than a weak number that looks strong.

The same discipline applies as with the four indicators. Ask about one concrete instance rather than the general case — *what did you do differently last week because of it* produces evidence, *has it been valuable* produces politeness. Ask the person who experiences the outcome, not the person who commissioned the work, because the commissioner's answer is partly about their own judgement and everyone knows it. And ask often enough that no single conversation carries too much weight.

Five real conversations with people outside the organization will tell you more than a survey with four hundred responses and a mean of 3.8. The survey is more defensible in a meeting. It is much less likely to change anyone's mind, which by the fourth rung is the only thing a measure is for.

Two failures worth naming

The first is the proxy. An organization decides to measure outcomes, finds outcomes hard, adopts something countable in their place, and within a year is optimizing that with the same efficiency it previously applied to output. Creating Value covers this at length; the short version is that the defence is not a better proxy but keeping one channel open that cannot be optimized — someone who talks to real users often enough to contradict the numbers with a specific human being.

The second is attribution. A great deal of energy gets spent trying to establish which team's work caused which improvement, and it is nearly always wasted, because the honest answer is that several things happened at once and the effect

belongs to all of them. Measure the outcome where it is actually visible — at the level of the service, the customer, the process — and accept shared credit. An organization that insists on attributing value to individual teams will end up measuring only the things small enough to attribute, which are reliably the things that matter least.

What to expect

The first reading is worth very little. There is no benchmark for any of this, nothing has been validated across enough organizations for anyone to say what a good number looks like, and the value of the measure is entirely in the comparison against your own earlier one. The fourth reading is worth a great deal.

Expect the early answers to be disappointing, and expect that to be the useful part. The first time an organization genuinely asks whether last year's work changed anything, the answer is usually *less than we said at the time*. That is not a sign the measurement is broken. It is the first accurate thing the reporting has produced in a while, and everything worth doing next follows from it.

EDITOR'S NOTES

Outcome over output is old ground and this chapter claims none of it. Melissa Perri's *Escaping the Build Trap* is the most useful modern statement of it, and the habit of writing down what you expect before the work starts and returning to it afterwards is hers — she calls it the product kata, and it is a discipline rather than a template. Anyone who wants the practice properly should go there. Outcomes and Key Results have been arguing the same case for longer, with mixed results, mostly because the mechanism gets adopted and the honesty does not.

The four friction indicators in Observing Organizations are diagnostic prompts rather than validated metrics, and this fifth one is weaker still — value resists definition in a way that waiting time does not. What this chapter contributes is the arrangement: four measures pointing inward at movement, one pointing outward at effect, and the observation that an organization can be excellent at the first four and learn nothing, because nothing in them can tell you whether the thing that moved was worth moving.

CHAPTER 53

Where to Start

Four moves small enough to make without asking permission, and one that isn't.

BOTH · WORKS ON ALL FOUR

If you have not already, start with A Closer Look.

Eighteen questions about what you can actually observe where you work, about seven minutes, no score and no grade. What it gives you is narrower and more useful than this chapter: the three or four chapters that match whatever is currently creaking, rather than a general answer to a general question. Working out which parts of a paper this size are yours should not be your job.

Then come back here, because knowing where to look and knowing what to do on Monday are different problems.

This is the one chapter in the paper allowed to sound like a checklist, and it should, because a reader who is convinced and doesn't know what to do next was lost for an avoidable reason.

This week. Pick one capability that matters. Ask three people in different parts of the organization who owns it. Don't announce that you're doing it, and don't correct the answers — just collect them. Whatever you find is the most accurate diagnostic in this paper.

This month. Time the next significant cross-team decision. Record when it became necessary, when it was made, and how much of the gap was deliberation. Bring both numbers to a leadership meeting without a recommendation attached. The number does the arguing.

This quarter. Take one thing you are currently funding as a project and describe it as a capability instead — what must we always be able to do, and is it better than last year? Notice how much of your reporting stops working when the question changes. That difficulty is the finding.

This year. Take one boundary — between two teams, two departments, two systems — and ask what it costs in handovers, waiting, and rebuilt context. Then ask whether the boundary is where it is for a reason that still holds, or because of a decision made in a room a long time ago by people who were solving a different problem.

And one that isn't small, if the authority happens to be yours. Take something currently funded as a project with an end date, and fund the capability behind it for two periods instead. That is a single decision, it cannot be delegated downward, and no amount of good work further down substitutes for it — a team can be perfectly clear about who owns an outcome and still be dissolved on schedule before the outcome arrives. Funding What Doesn't End is the long version.

The first four are deliberately small enough to do without permission. This one is deliberately not, and if you are reading the other four and concluding they are somebody else's task, this is the one that was written for you.

None of these are transformations. That's deliberate. The first four are small enough to do without permission, and each one produces a fact rather than an opinion — which, in an organization where the friction is invisible, is the only thing that reliably changes anyone's mind.

What to do with what you find

Collecting the fact is the easy half. What happens next decides whether any of it was worth doing, and there are three ways it usually goes wrong.

Don't fix it quietly. The temptation, having found that nobody owns something, is to sort it out yourself — because you can, and because raising it feels like making trouble. Do that and you have removed the evidence along with the symptom. The organization learns nothing, and the same gap reappears somewhere you aren't.

Don't present it as an indictment. Every number you collect has people attached to it. The decision that took eleven weeks was waited on by someone who could have unblocked it, and that person is likely in the room. Present the wait as a property of the system, because it is one — nobody designed the eleven weeks, and nobody chose them.

Don't propose the reorganization. The instinct after finding unclear ownership is to redraw boundaries, and it is almost always premature. The first move is to make one existing owner unambiguous and see what stops being needed. Structural change is expensive, slow, and burns the credibility you will need for the second finding.

How to put a number in front of people

The numbers in this paper are small, personal, and easy to dismiss. Presented badly they sound like a complaint with arithmetic attached. A few things make the difference.

Bring the distribution, not the average. One decision that took fourteen weeks tells a room more than a mean of five, because everyone present can remember the fourteen.

Attach it to something already agreed. Nobody funds "reducing decision latency." They do fund the initiative that is late because of it. The number is a diagnosis of something the organization already cares about, and should be introduced that way.

Say what it would take to be wrong. Volunteering the weakness of your own measurement is the fastest route to being believed. It is also honest: these are small samples, self-reported, and directionally useful rather than precise.

Ask for the second measurement, not for the change. The first number proves nothing on its own. Getting agreement to take it again in three months is a much smaller ask than getting agreement to act, and it commits the organization to noticing.

What to expect

Less than you want, later than you'd like.

The first measurement will be dismissed by someone as unrepresentative, and they will have a point. The second one will be dismissed too. Somewhere around the third, the conversation changes — not because the evidence became overwhelming, but because a number that keeps coming back stops being an argument and starts being a condition.

That is the honest timescale, and it is worth saying plainly rather than promising a workshop that fixes it. Organizations change at the speed of the loop they can actually close.

What it costs you

No chapter in this part has said what any of this costs the person doing it, and a paper that asks people to work differently without pricing it reads as free, and therefore as unlikely.

It costs visibility.

The evidence that this is working is an absence. A meeting that stopped happening. A wait nobody experienced. An escalation that never arrived, a decision that did not need you,

an incident that did not occur. Absences do not report themselves, do not appear on a scorecard, and cannot be described afterwards without sounding like a claim about a hypothetical.

So anyone doing this well will have a quieter quarter than a colleague who staged a visible rescue, and will be less legible in the room where those things get discussed. That is the actual trade, and it is worth knowing about in advance rather than discovering it in March.

I am not going to dress that up. It is the same pattern No Incident, No Story describes, applied to the person reading this — and the only honest mitigation is to be deliberate about the record: write down what you expected before you change anything, so that later you have a before and an after rather than a feeling.

The compensation, such as it is, is that this work is unusually durable. The forum you removed stays removed. The ownership you settled tends to stay settled long after anyone remembers it was ever unclear, which is both the reward and the reason nobody will thank you for it.

So start where you can already see something. What is hidden further in tends to become visible once the first thing moves, and rarely before.

CHAPTER 54

What Goes Wrong

The five ways this gets applied badly, including by me.

BOTH · WORKS ON ALL FOUR

Every idea about organizations has a characteristic way of failing, and it is usually a distorted version of its own best feature. Worth naming the ones this paper has, because a reader who recognizes the failure mode in advance has a decent chance of avoiding it, and because I have produced most of them personally.

It becomes a vocabulary

The fastest one, the least visible while it is happening, and the one I have helped along most often — usually by being pleased that people had started using the words.

Words are the cheapest part of any idea to adopt. Within a quarter of introducing this language somewhere, people are saying *capability* where they used to say *system*, *guardrail* where they used to say *rule*, and *flow* where they used to say *speed*. Nothing else has changed. The same decisions travel the same distance to the same forum, now described in a more fashionable register.

This is worse than not adopting the idea at all, because the vocabulary provides cover. An organization that says it works this way is harder to persuade than one that has never heard of it.

The tell is that no meeting has been cancelled. Real adoption removes things. If six months in, the calendar looks the same and the escalation paths are unchanged, what was adopted

was a glossary.

The diagnosis becomes a scorecard

The four indicators exist to open a conversation. Attach a target to any of them and the conversation closes — and I have been the one attaching it, because a number somebody is accountable for is easier to take into a steering group than a number somebody is merely curious about.

Decision latency is the easiest to ruin: it improves immediately if you decide faster and worse, and there is no term in the measurement for whether the decision was any good. Ownership clarity is close behind — one memo naming owners moves the number without moving anything real. Learning cycle time can be gamed by holding a retrospective.

I have watched a perfectly good diagnostic become a quarterly reporting obligation, at which point it stopped measuring the organization and started measuring the reporting. The indicators are for the people doing the work, in service of a conversation they choose to have. Roll them upward and they will be optimized directly, which is what happens to every organizational number that acquires an audience.

Autonomy is granted without guardrails

The paper argues for pushing decisions to where the information is. Read quickly, that sounds like removing constraints, and removing constraints is a thing leaders can do in an afternoon and feel good about. I have done exactly that, believing I was giving people room, and given them the job of guessing what I would have wanted instead.

What follows is a period of enthusiasm, then a period of divergence, then a rediscovery of why some of those constraints existed. Six months later authority is quietly recen-

tralized, and the organization has learned the wrong lesson: that autonomy was tried and did not work.

Autonomy without settled boundaries isn't ownership. It's abandonment with better branding, and the recentralization that follows costs more credibility than the original state ever did. Guardrails first, then authority. In that order, every time.

It gets sold as a transformation

This is the one I would be most likely to do again, because it is how you get funding and I have wanted the funding. The moment it becomes a programme, it acquires a name, a steering group, a slide deck, and a completion date — and every one of those works against the thing it is meant to deliver.

A steering group is a decision-making body created to oversee the reduction of decision-making bodies. A completion date is a claim that conditions are a project. The name means that from then on, every improvement has to be attributable to the programme, which quietly kills the improvements nobody can attribute.

The moves in Where to Start are deliberately small enough to make without permission. That constraint is doing real work. An improvement nobody had to fund is an improvement nobody can cancel.

The gardener gets impatient

This is my own failure mode, and the one I now recognize fastest, because I have been in it.

Cultivation is slow, and the organization is asking for a plan. So you keep the language of conditions and revert to the behaviour of instruction: shaping the conditions in the morn-

ing and intervening on outcomes by the afternoon, because a specific thing is going wrong and you can see exactly how to fix it. Every individual intervention is defensible. Together they teach the organization that autonomy holds until it matters.

I have taken back a decision I had explicitly delegated, for reasons that seemed excellent at the time, and watched the effect on how much the team decided afterwards. It is not subtle. One retraction is worth more, as a signal, than ten delegations — because it is the one that tells people what the actual rule is.

The discipline is to let a bad outcome stand when the alternative is teaching the room that the delegation was conditional on being right. That is the hardest thing in this paper to actually do, and I would not claim to be reliable at it.

The pattern underneath all five

Each of these is the organization converting something that requires patience into something that can be reported this quarter. A vocabulary, a scorecard, a programme, a memo, an intervention — all legible, all fundable, all immediate.

Conditions are none of those things. That is not a flaw in the argument; it is the whole reason the friction persists in organizations full of intelligent people who can see it perfectly well. The work is unglamorous, it is slow, and most of the evidence that it worked is an absence — a meeting that no longer happens, a wait nobody experienced.

An organization capable of valuing that has already solved most of what this paper describes.

CHAPTER 55

Working This Way

What it actually looks like to run an organization through this lens, week to week.

BOTH · WORKS ON ALL FOUR

If this paper has done its job, it hasn't given you another framework to implement.

It has given you another way to see — and a way of working that follows from it, which is the part worth being concrete about, because a lens you cannot act through is just a nice sentence.

So this chapter is about the practice. What changes in an ordinary week, from the two vantage points this paper is most often read from — the one that sets the conditions, and the one that works on them directly.

If you run the organization

Four habits, and none of them require a mandate, a programme, or anyone's permission.

Ask where it waited, not why it was late. The standard executive question — *why is this taking so long* — reliably produces a defence, because it is heard as an accusation about effort. The same information comes out of a different question: between the moment this was decided and the moment it reached a customer, where did it sit still, and for how long? That question is answerable, nobody has to protect themselves from it, and the answer is almost always somewhere nobody expected.

Fix ownership before you fix process. When something is going badly, the available moves are usually a new forum, a new process, or a new report. Before any of those, establish whether one person can say *I own this and I decided*. If they can't, every process you add is a compensation, and you will be maintaining it for years.

Count what you removed. Most executive scorecards only go up. Keep a private list of the meetings that stopped happening, the approvals that were retired, the decisions that no longer come to you. If that list is empty after two quarters, nothing structural has changed, whatever else has improved.

Protect one channel to the outside that nobody can optimize. Someone who talks to real customers often enough to contradict the dashboard. This is the first thing cut when the quarter gets tight, and it is the only defence against becoming extremely efficient at the wrong thing.

If you work on the organization's ability to change

This is a competence rather than a job title, and the people holding it are not always called architects. They are operating-model people, transformation leads, engineering and product managers, chiefs of staff — anyone whose actual subject is how the organization decides and how easily it can change its mind. Architects are among them, and the work described here is not the one on most architecture job descriptions, which is worth stating plainly.

Spend your time on boundaries, not on artifacts. A diagram, a target operating model, a strategy deck — each is a way of thinking, and occasionally a way of explaining. None of them is the work. The work is where the lines fall, who is allowed to decide what, and which capability is currently split across two departments that both believe the other owns it.

Turn up where the architectural decisions are actually made. They are made in budget rounds, reorganizations, hiring decisions and vendor selections — mostly by people who were told architecture was someone else's job. Anyone responsible for the organization's ability to change has to be present in those rooms, not only in the technical forums.

Settle a guardrail rather than review a decision. Every review you perform is a decision that had to travel. Every guardrail you settle removes a whole class of them permanently. Reviewing is more visible and more immediately satisfying, which is why most architecture functions do too much of it.

Say what you would need to see to be wrong. Arguments about structure become religious wars precisely when nobody has stated what evidence would settle them. Volunteering the conditions under which you would change your mind is the fastest way to be believed, and it costs nothing you were actually using.

Measure something, even badly. Four hours of handoff loss, self-reported by one team, is worth more in a leadership conversation than any model you will ever draw — because it is arithmetic, and arithmetic can be argued with.

If you sit in the middle

Both sections above are written for people with a mandate. The person most trapped in the friction has neither, which is why this section exists.

Middle management is the only place in an organization where the paradox is fully loaded. You are rewarded on headcount, in most places still expected to be the person who decides, and structurally required to route decisions upward through people who hold less detail about them than whoever raised them. Every one of those is somebody else's design, and all three land on you.

It is also, and this is the part that gets missed, the only position from which decision authority can actually be handed downward. An executive can announce that teams are empowered. Only the person the decision currently routes through can stop being in the route.

Push one decision down and hold your nerve when it goes badly. The same move as elsewhere in the paper, and considerably harder here, because you will be asked about the outcome by somebody who does not know you delegated it. Answering *that was the team's call and I stand behind it* is the whole practice, in one sentence, and it is expensive exactly once.

Stop being the translation layer. Much of what reaches you is a question that would be answered better by connecting two people directly. Every time you relay instead of introduce, you have made yourself load-bearing for a conversation that did not need you — and No Incident, No Story explains why nobody will notice you stopped.

Say what the structure makes hard, upward, before it is announced. You can see the three things a reorganization will slow down, and the room deciding it usually cannot. That information has a short window: raised beforehand it is design input, raised afterwards it is resistance. Same sentence, different reception.

And protect your own list. You are also the place where waiting accumulates, because everything routes through you. Ten Minutes Today applies to you more than to anyone else in the organization, and the answer to *who is waiting on me* is longer here than it is anywhere else.

None of that is a complaint about managers, and none of it is sympathy. It is a description of the one seat that can hand authority downward, held by people who are usually measured on holding onto it.

What all three vantage points share

The discipline underneath all three is patience with things that do not report well.

Conditions are slow, mostly invisible, and their evidence is an absence — a meeting that no longer happens, a wait nobody experienced. There is no way to make that legible on a quarterly slide without turning it into something else, which is exactly the failure mode the previous chapter describes.

So the work is largely unwitnessed, and the people doing it get very little credit, and it compounds anyway. That is the trade. An organization capable of valuing it has already solved most of what this paper describes.

CHAPTER 56

Seeing It Where You Are

The examples were never meant to be copied. They were meant to make something visible.

The examples throughout these chapters come from my own experience, but they were never meant to be copied. They were meant to make something visible.

My hope is that, wherever you work, you begin to notice the places where value creation slows without anyone intending it to. Where information stalls. Where decisions travel too far. Where ownership becomes blurred. Where friction quietly accumulates.

And I hope you also begin to notice the opposite.

The moments where work seems almost effortless. Where decisions are made close to the problem. Where people understand what they own. Where learning changes what happens next. Where the thing being built is plainly worth building.

The stories in this paper are only examples.

The real work begins when you start seeing your own organization, your own products, your own teams, and your own decisions through this lens.

If this paper helps you notice flow and friction where you are — and encourages you to improve one small part of it — then it has accomplished exactly what I hoped it would.

The friction that nearly stopped this

For a long time the greatest friction for this paper was finding the time to turn years of ideas, notes and experience into something anyone else could read. Writing, editing, publish-

ing, building a website, preparing formats — all of it demanded skills that had nothing to do with the ideas themselves.

That kind of friction has almost disappeared. The distance between having something to say and being able to share it is close to gone, and the same thing is happening to the distance between an idea and a working system. For decades that gap required deep technical knowledge. It increasingly doesn't.

Which makes the ideas matter more, not less. The tooling will build very nearly anything you can describe — but you still have to have the idea, or know the problem well enough to recognize what is actually worth building, and then cultivate it into something that holds. That part hasn't moved at all.

And the friction that costs organizations most was never that kind anyway. It doesn't live in one place — it's everywhere and nowhere at once: a little in the handoff, a little in the approval, a little in nobody being quite sure who decides. Which is exactly why technology can't remove it. Any single step can be made faster, but the friction was never in the steps. It was in the seams between them, and making the steps faster doesn't make the seams disappear. It just makes it more obvious that they are still there. Seeing those seams, and cultivating the conditions that close them, is still entirely on us.

The material in this paper is mine. The tooling only helped me share it.

Somewhere along the way I stopped seeing the future of architecture as designing the business, and started seeing it as gardening the business — cultivating the conditions for what it needs to become. I may be a little biased toward that metaphor by now.

Everything here is written about the professional side of your life, because that is where I have the standing to write it. But friction and flow are not unique to organizations. They

show up at home, between friends, in how a family makes decisions or quietly lets one person carry too much. I have deliberately left that door closed in this paper.

But think about it anyway.

About the Author

I have spent more than twenty years in technology, architecture and organizational change. I wrote my first production code in 2001 for an insurance company most people have never heard of, moved from writing systems to designing them, and then from systems to the domains around them — solution architecture, domain architecture, and eventually Chief Architect and Head of Architecture, leading the function from the management team while still working hands-on.

Since then I have worked across a number of organizations as an architect and adviser, which turned out to be the useful part. Seeing the same mechanisms in places that had nothing else in common is what made me trust them.

The throughline, in hindsight, is a slow move away from the code and toward the conditions around it. Organizations with comparable people, comparable technology and comparable intentions kept producing very different outcomes. The technology explained some of that and rarely all of it, and what was left over is what this paper is about. Organizational Flow is the name I eventually gave to it.

None of what follows is research. It is an accumulation of observations from practice, written down because the patterns kept repeating, and offered as a working theory rather than a finished one — deliberately unfinished, and better every time somebody argues with it.

Acknowledgments

Ideas rarely arrive alone.

This paper is the product of twenty-five years of practice, reflection, conversations, and shared learning.

My first and greatest thanks go to Anki, and to Isa, Tilde and Oliver.

You are the best part of my life.

Thank you for your patience, your love, and your unconditional support.

My thanks also go to Lars Barkman, whose conversations became the starting point for many of the ideas in this paper, and whose feedback on the drafts improved them.

And to David Pettersson, who read this properly and sent back the kind of notes that are only worth having from someone who took it seriously. Several things in this version are here because he raised them.

And to everyone I have had the privilege of working alongside over the past twenty-five years — colleagues, clients, and friends. You gave me perspectives I didn't have, challenged ideas I held too tightly, pushed me further than I would have gone on my own, and trusted me with opportunities long before I felt I had earned them.

What follows are the books, research, ideas and practices I took inspiration from, and where I went looking to see whether what I thought I was noticing had already been no-

ticed — and noticed more rigorously. Some confirmed what I had come to on my own. Some corrected it. They are acknowledged where their influence is most visible, rather than gathered at the back as references.

Drive: The Surprising Truth About What Motivates Us

DANIEL PINK

The autonomy, mastery, and purpose framework this paper leans on for what makes people thrive at work — and what turns out to map onto gardening almost too neatly.

Appears in: Growing the Team.

Self-Determination Theory

EDWARD DECI & RICHARD RYAN

The research underneath the autonomy, mastery, and purpose framework. Pink made it readable; Deci and Ryan did the work, and anyone wanting the evidence rather than the story should start with them.

Appears in: Growing the Team.

Thinking in Systems

DONELLA MEADOWS

The clearest available argument that systems behave according to their structure rather than anyone's intentions for them. The forest in this paper is its own image, but this is the thinking underneath it.

Appears in: It Grew Like That.

General System Theory

LUDWIG VON BERTALANFFY

The original case that living systems obey principles no machine model captures — decades before anyone applied it to organizations, and the lineage every version of this idea descends from, including this one.

Appears in: It Grew Like That.

The Fearless Organization

AMY EDMONDSON

Psychological safety as a concept, and the research establishing it. This paper argues that safety follows from clear boundaries rather than running alongside them, which is a claim about sequence — the underlying idea is hers.

Appears in: A Meeting Grew Here.

Sociocracy, Holacracy & Open Space

PRACTICES THIS PAPER BORROWS FROM WITHOUT ADOPTING

Self-selection did not start in the room described in Ownership. These traditions worked it out first, and the chapter takes one narrow lesson from a much larger body of practice.

Appears in: A Meeting Grew Here.

Domain-Driven Design

ERIC EVANS

Bounded contexts are the closest existing relative to what this paper means by capabilities — drawing lines around what belongs together, and treating the lines rather than the systems inside them as the durable thing. The wider enterprise-architecture tradition is in here too, less as a single source than as the field where I learned which questions were worth asking.

Appears in: What Survives the Reorg.

Team Topologies

MATTHEW SKELTON & MANUEL PAIS

Team types, interaction modes, and cognitive load as a sizing constraint rather than headcount. Anyone drawing team boundaries should read the original rather than the summary of it in this paper. What I add is the argument about decision authority — that embedding business judgement in a team, rather than representing it, is what separates a team that can move from one that can only build.

Appears in: The Team That Has to Ask.

Escaping the Build Trap

MELISSA PERRI

The case that the project is the wrong unit, that the funding model produces the behaviour rather than the people, and that writing down what you expect before the work starts is what makes being wrong useful. She makes the argument more thoroughly than I do, including the part above the team that this paper skips. What I add is the friction reading — that a project assembles boundaries, a steering forum and a dissolution date on purpose, and so manufactures three of the four shapes before anyone has done any work.

Appears in: Funding What Doesn't End.

Building Evolutionary Architecture

NEAL FORD, REBECCA PARSONS & PATRICK KUA

The clearest statement of the standard this paper's Architecture chapter uses — that an architecture's first job is to remain guided and changeable — and the operationalization I don't have, in the form of fitness functions. I apply the same standard one level out, to organizational decisions, where nothing can be automatically verified. Anyone applying it to systems should read them for the mechanics.

Appears in: The Long Way to a Yes.

Documenting Architecture Decisions

MICHAEL NYGARD

The ADR format, unchanged since 2011 and still the only architecture documentation I care about keeping. The argument in Settling the Few Things for where decision records sit is mine; the thing itself is his.

Appears in: Settling the Few Things.

Technology Radar

THOUGHTWORKS

Adopt, trial, assess, hold — a format that makes technology decisions explicit, dated and reversible. Its real value is not the published list but the forum an organization has to create in order to produce its own.

Appears in: Faster, and Still Wrong.

Accelerate

NICOLE FORSGREN, JEZ HUMBLE & GENE KIM

The measured research behind software delivery performance, and the reason this paper deliberately stops where it does. The four keys are properly operationalized and statistically validated; the four indicators here are diagnostic prompts sitting one layer upstream, and nothing in this paper carries the same evidential weight.

Appears in: Faster, and Still Wrong.

The Infinite Game

SIMON SINEK

The framing of organizations that treat learning as continuous because there is no finish line to reach — which turns out to describe the same posture that attracts the people worth keeping.

Appears in: The Wait Nobody Logged.

Version History

This paper is not finished, and it isn't meant to be.

2026-06-30

Where this started. The site was rebuilt, the blog archive moved across with its original links intact, and the paper begun. What you are reading now is the First Public Draft — it has changed a good deal since, and it will keep changing. I have stopped listing each revision here; the draft status on every chapter says what it is, and the paper is more useful open and evolving than it would be finished and quiet.

Also Online

Some of this paper does not belong in a book, and is better where it can keep changing. It is all at troi.se/organizational-flow, free.

A Closer Look — eighteen questions about what you can observe where you work, which then points you at the chapters that match. It needs a browser to do its arithmetic, so it is not in these pages. troi.se/organizational-flow/self-assessment/

The narration — every chapter read aloud, and the whole paper end to end, for a run or a drive. troi.se/organizational-flow/print/

Principles — troi.se/organizational-flow/principles/

Further Reading — troi.se/organizational-flow/further-reading/

Print Edition 1.0. Based on the online edition as of 13 August 2026. Organizational Flow is a living paper that continues to evolve online. This print edition represents a snapshot of the text at the date shown above. The current edition is at https://troi.se/organizational-flow/.

AI Transparency

Short version: the thinking is mine, the typing had help. Since the paper spends a chapter on what AI does and doesn't do for an organization, it would be a bit rich not to say how I used it here.

What I actually did

The observations in this paper come from about twenty-five years of working inside organizations and noticing the same things happening. I carried most of it around unwritten for a long time. Getting it out of my head and into something anyone else could read is the part that took until 2026.

I wrote with Claude, and it earned its place. It argued with me, which was the useful bit. It caught the places where I had said the same thing twice in different words, asked what I meant when I hadn't decided yet, and found the paragraph where I contradicted something I had written eleven chapters earlier. A good editor does that. This one is available at two in the morning.

What it did not do is have the ideas. It has never sat in a steering group watching a decision take three weeks, or run a function, or been wrong about an architecture in a way that cost somebody money. Every observation here is one I have made myself, and every judgement about what to keep, cut, soften or stand behind is mine.

There is a version of this note that lists tools and process steps and reassures you about editorial control. I would rather just say it plainly: the ideas are mine, some of the

sentences are better than they would have been on my own, and I read every one of them before it went up.

Organizations with similar people, similar technology, similar money and similar ambitions produce remarkably different results. That gap is what this paper is about.

Somebody sends a message on Tuesday morning. The answer comes Thursday afternoon. Nobody did anything wrong, and two days is a perfectly civil response time. Now multiply that by everyone currently waiting on somebody else, which is very nearly everyone.

Organizational Flow is the property underneath that: how easily information, decisions, execution and learning actually move through an organization. It rises when that movement is easy and ownership is clear. It falls when structural friction accumulates faster than value is created.

The mechanisms are rarely dramatic. Someone is not sure whose decision this is, so they check with somebody first. Checking becomes a standing meeting, and the meeting becomes how things are done here. Every step was reasonable, and nobody would defend the result.

Which leaves a question worth asking more often than it gets asked: how much more could this organization create with exactly what it already has?

Christoffer Råsten has spent twenty-five years in technology and architecture, most of it noticing why comparable organizations move at such different speeds.

troi.se/organizational-flow