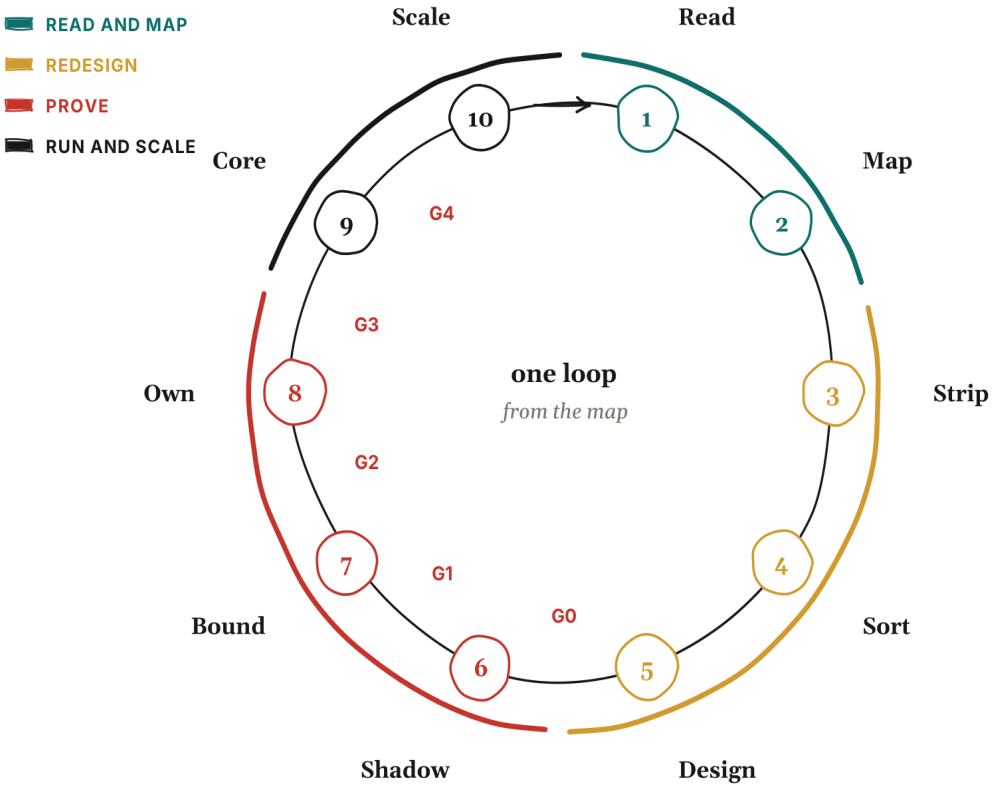
The Embedding Lifecycle

Ten Steps From a Working Pilot to AI in the Enterprise Core



> Read before you build. Prove before you act.
> Sign every gate.

Gennaro Cuofano

The Embedding Lifecycle

Ten Steps From a Working Pilot to AI in the Enterprise Core

The Embedding Lifecycle
Ten Steps From a Working Pilot to AI in the Enterprise Core

The Business Engineer Library · The Embedding Series, Volume III

Gennaro Cuofano
businessengineer.ai

First published 2026. The composite manufacturer in this book is illustrative. Its figures are authored to teach the method, set within published ranges, and not drawn from any company. Sources are listed in Notes and Sources.

Figures drawn for this volume in the Library's ink register.

Contents

The Middle 6

Part One: Ground **9**

I. What Embedding Means 10

II. The Executor That Decides 14

III. What the Executor Is Made Of 17

IV. The Words This Book Uses 23

V. Lifecycles Before This One 30

VI. The Lifecycle at a Glance 35

VII. Why the Lifecycle Has This Shape 39

VIII. Before Step One 44

Part Two: Read and Map **47**

IX. Step 1: Read 48

X. Step 2: Map 54

Part Three: Redesign **60**

XI. Step 3: Strip 61

XII. Step 4: Sort 67

XIII. Step 5: Design 74

Part Four: Prove **80**

XIV. Step 6: Shadow 81

XV. Step 7: Bound 86

XVI. Step 8: Own 91

Part Five: Run and Scale **97**

XVII. Step 9: Core 98

XVIII. Step 10: Scale 104

Part Six: Across the Lifecycle **110**

XIX. How Lifecycles Break 111

XX. Adapting the Lifecycle 118

XXI. Running Many Loops at Once 123

XXII. Questions Leaders Ask 127

XXIII. What Finishing Is Worth 132

The Cycle Closes 136

Appendix A: The Lifecycle on One Page 138

Appendix B: Who Signs What 139

Appendix C: Entry and Exit Checklists 140

Appendix D: The Case in Numbers 142

Appendix E: The Lifecycle and Governance Frameworks 146

The Five Moves 147

The Meters 148

The Mental Models 149

Glossary 151

Notes and Sources 153

The Thread 156

Also in This Series 157

The Middle

Most companies can now show you an AI pilot that works. A model reads supplier invoices and proposes the right cost centre. An agent drafts replies to customer emails, and the drafts are good. A demo runs on a laptop in a meeting room, and the room is impressed.

Far fewer companies can show you the same work running inside the business a year later. By that we mean deciding real cases, writing to the systems of record, and producing results that someone has signed for.

The gap between those two pictures is the subject of this book. We will call it the middle.

The numbers on the middle are consistent, even when the surveys disagree on almost everything else. In 2025, an MIT study of enterprise AI programmes reported that only about 5% of custom enterprise tools reached production. McKinsey's survey that year found that 88% of organisations used AI in at least one function. Only about a third had begun to scale it, and no more than 10% were scaling agents in any single function. Gartner predicted that over 40% of agentic AI projects would be cancelled by the end of 2027, citing costs, unclear value and weak risk controls.

These figures are not a verdict on the technology. A pilot is a narrow test. It shows that a model can do a task on a prepared set of cases. It does not show who answers when the model is wrong, which rules it follows when the policy and the system disagree, what it costs per accepted result, or how anyone would switch it off.

Those questions are not technical. They are questions about the business. They also have to be answered in a particular order. You cannot decide who signs off a large invoice before you know which decisions the work contains. You cannot measure savings before you have measured the baseline. And you cannot measure the baseline before you know what the work really is.

Most stalled programmes did not fail at the model. They failed in the middle, because nobody owned the sequence that turns a working demo into work the business relies on.

This book describes that sequence. It is a lifecycle of ten steps, grouped into four phases. Each step answers one question, produces one or more documents that a named person signs, and ends at a gate that says whether the

work may move on. The steps run in order because each depends on the answers of the one before.

There is a deeper reason the order matters. An embedded loop combines two kinds of machinery that have been studied for a long time. One learns from examples: the model, flexible and strong on routine cases, but without guarantees. The other is written down: rules, decisions, limits and standards, precise and auditable, but never complete. The model can be bought. The written half cannot, because it describes this company's own work. The lifecycle is the sequence in which a company builds that written half, tests it against real cases, and closes the loop so that results feed back into the design.

The lifecycle is built for one particular kind of change: putting an executor that makes decisions inside a company's core processes. That is different from giving staff a writing assistant, and different from installing software that follows a script. When the executor decides, the design of the work has to say what it may decide, where a person decides instead, and how anyone would know the decisions were good. The lifecycle exists to get those answers written down and signed before the executor acts alone.

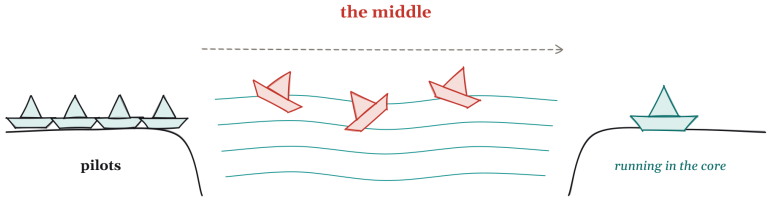
The book is written for the people who carry that responsibility. That includes the executive who sponsors the change and the owner of the process being changed. It also includes the finance, risk and operations people who must agree to it, and the practitioners who run the work. No technical background is assumed. Where the book uses a term, it explains it first.

A single composite company runs through every chapter. It is a mid-size European maker of industrial pumps and valves, with about 4,200 people and three plants. We follow one of its processes, the approval of supplier invoices, from the first reading of where value sits to the day it runs in the core. The company and its figures are illustrative. They are set within published ranges, which the notes identify, so that the arithmetic teaches something real.

Part One lays the ground. It covers what embedding means, why an executor that decides changes the design of work, and what that executor is made of. It also defines the handful of terms the rest of the book depends on, looks at what earlier lifecycles got right, and explains why the lifecycle has the shape it has. It ends with what must be in place before the first step. Parts Two to Five take the ten steps in order. Each step closes with what the foundations add: the reasons, drawn from logic, learning and control, that the step works as it does. Part Six covers how lifecycles break, how to adapt them to different kinds

of loop, and how a company runs many at once. It also answers the questions leaders ask, and shows what finishing is worth. The appendices hold the lifecycle on one page, the signatures step by step, the checklists, and the full case in numbers.

Read it in order the first time. After that, it is a reference you can open at whatever step your own work has reached.



Like a wide river: *many pilots launch from the near bank, few cross the middle, and only those that land do real work for the business.*

PART ONE

Ground

I

What Embedding Means

Companies have used software for sixty years, and AI for more than ten. So it is fair to ask what is new about embedding it now.

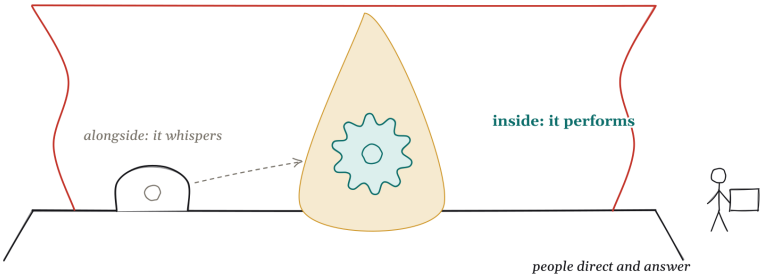
The answer starts with a distinction between two places a technology can sit. It can sit alongside the work, helping the people who do it. Or it can sit inside the work, doing part of it.

A spell-checker sits alongside the work. So does a search tool, a dashboard, or a chat assistant that drafts an email a person then edits and sends. In each case a person remains the executor. The person reads the result, decides what to do with it, and acts. If the tool is wrong, the person catches it or does not, and either way the person answered for the outcome.

Embedding means putting the technology inside the work. The executor reads the invoice, checks it against the order, decides whether it can be approved, and posts it to the ledger. Nobody retypes or approves each result. People still design the work, set its limits, decide the hard cases and answer for the outcome. They no longer touch every case.

The difference matters because the alongside position carries almost no organisational risk. If a writing assistant drafts a poor paragraph, a person deletes it. Nothing in the company's records changes. That is why assistants spread quickly: they ask nothing of the company's processes, its controls, or its accountability.

Embedding asks for all three. An executor inside the work writes to systems of record. It commits the company to suppliers, customers and regulators. Its mistakes become the company's mistakes, recorded in the ledger. So before it may act, the company has to know what it is allowed to do, under which rules, with which evidence, and who answers for it.



Like moving from the prompter's box to the stage: *alongside, the tool whispers lines to a person; inside, the executor plays the part while people direct it.*

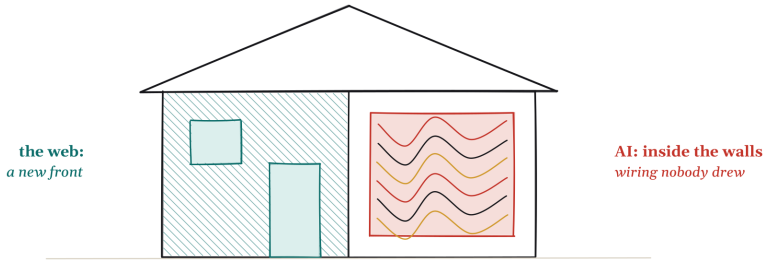
Outside in, inside out

A useful way to see why this is hard is to compare AI with the previous great wave of technology, the web.

The web worked from the outside in. It changed how companies reached customers first: the storefront, the catalogue, the marketing, the support pages. The back office changed later and more slowly, and often it did not change much at all. A company could put a modern website on top of thirty-year-old processes, and many did.

AI works from the inside out. Its value sits in the work itself, in reading documents, applying rules, making decisions and producing results. To deliver that value it has to enter the processes, and the processes are where companies are least documented. Rules live in policies, in system settings and in the heads of experienced staff, and the three rarely agree. Steps exist for reasons nobody remembers. Decisions that look routine turn out never to have been agreed.

This is why a pilot can succeed and the programme still stall. The pilot tested the executor on a clean slice of work. Embedding exposes the executor to the real process, with all its undocumented rules and buried disagreements. The lifecycle in this book is mostly a way of surfacing those things, in order, and getting someone to decide them.



Like renovating an old house: *the web repainted the front, but AI has to go inside the walls, where the old wiring was never drawn on any plan.*

What embedding is not

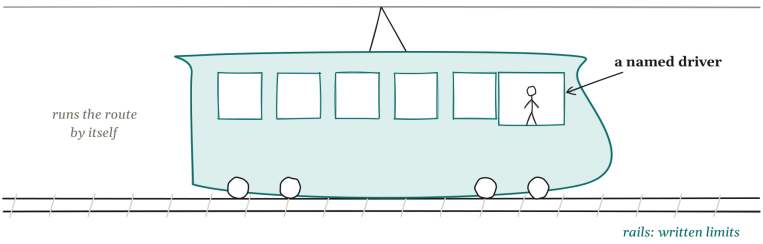
Three things are often confused with embedding, and it helps to set them aside.

Embedding is not deploying a tool. Licences can be bought and switched on in a week. A tool that staff may use is still alongside the work.

Embedding is not automating a process map. Most process maps record how work was done under old constraints. Attaching an executor to each box speeds up the steps but keeps every workaround, and it usually leaves the waiting between steps exactly where it was.

Embedding is not removing people. A well-embedded process has fewer people touching routine cases and more attention on the cases that matter. Roles change, and some disappear. But a process with no named person answering for it is not embedded. It is abandoned.

What embedding is, in one sentence: **an executor that decides runs part of a core process, inside limits the business has written down, with named people answering for the result.** The rest of this book is about getting there.



Like a tram: *it runs the route alone on rails laid in advance, with a named driver who can stop it; without one, it is a runaway.*

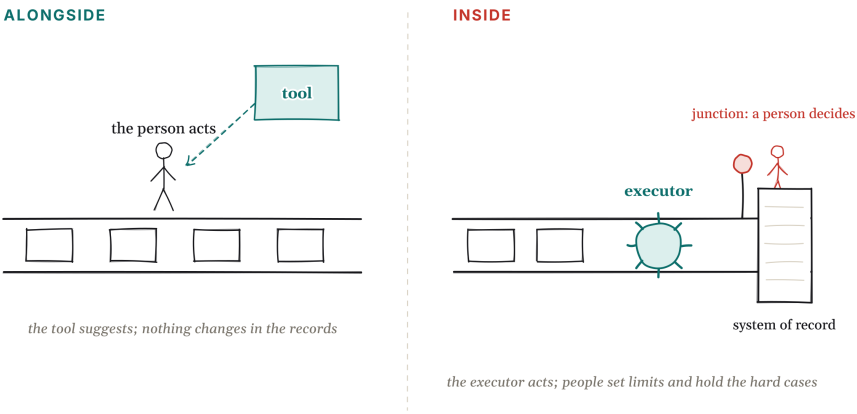


FIGURE 1 Alongside and inside. On the left, the tool helps and the person acts. On the right, the executor acts inside the work, people set its limits and hold the hard cases, and results land in the system of record.

II

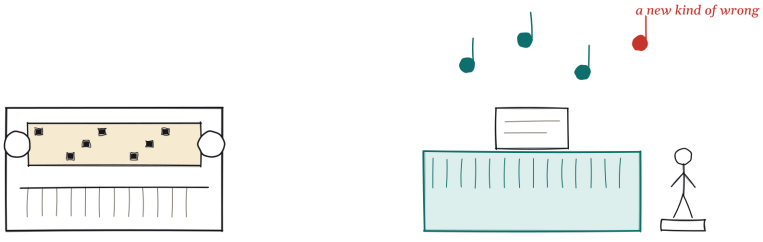
The Executor That Decides

For a century, companies designed work around executors that followed instructions. That assumption shaped every method they used, and it is the assumption that has changed.

Consider the executors a process could rely on before. A clerk followed a procedure manual. A machine on a line repeated a motion. An enterprise system applied the rules configured in its screens. A software robot clicked through a sequence of fields. Each of these needed the work to be specified in advance, step by step. Where the specification ran out, a person took over.

So process design meant writing instructions. The process map was a set of instructions for people and machines. Control meant checking that the instructions had been followed. Exceptions were whatever the instructions did not cover, and they went to a person.

A language model with access to tools is a different kind of executor. Given an outcome and some context, it can work out the steps itself. It can read a document it has never seen and decide what kind of document it is. It can apply a rule written in plain language to a case nobody anticipated. It can be wrong in ways no instruction-follower could be wrong, because it was never following an instruction in the first place.



Like a player piano beside a jazz pianist: *one plays exactly the holes in its roll; the other is given the tune, finds the notes, and can miss.*

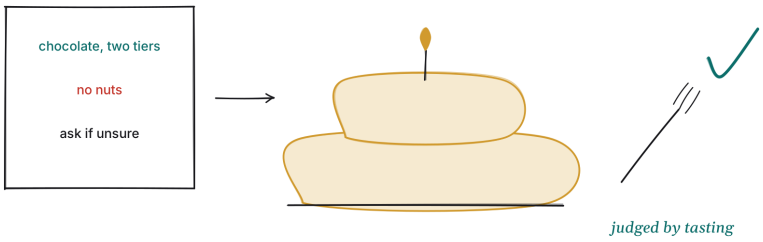
Three consequences

This changes the design of work in three ways, and each shows up later as a step in the lifecycle.

First, the unit of design moves from the task to the outcome. If the executor can find its own steps, describing steps becomes less important than describing the result, the rules and the limits. A task list tells an instruction-follower what to do. A decider needs to know what counts as done, what it may not do, and when to stop and ask.

Second, decisions become the thing to design. With instruction-followers, decisions were buried inside steps, and a person made them without anyone listing them. With a decider, every decision must be sorted. Some it can take alone. Some it can take within a written threshold. Some must always go to a person. And some, it turns out, the company has never actually agreed on.

Third, control moves from checking steps to grading outcomes. You cannot audit a decider by confirming it followed the procedure, because it did not have one. You audit it by grading what it produced, every result, against a written standard.



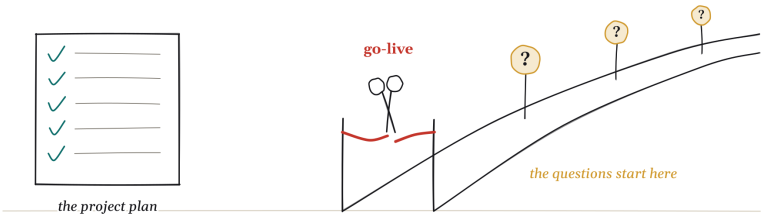
Like ordering a cake from a baker: *you describe the cake, say what it must not contain, and judge it by tasting, not by watching each step.*

Why the old methods stall

Most companies approach an AI programme with the methods they already have. These are a project plan, a process map, a requirements document and a testing phase. Each was built for instruction-followers, and each misses something a decider needs.

The process map records steps, but not which steps are workarounds. The requirements document lists functions, but not the decisions inside them or who owns each one. Testing checks that the system does what it was told, but a decider was not told; it was given an outcome. And the project plan ends at go-live, which is precisely where the questions about a decider begin. Who watches it? How is drift caught? Who may switch it off?

None of this means the old methods are useless. Several of their best ideas reappear in the lifecycle. It means they are incomplete for this executor, and the gaps sit exactly in the middle, between a pilot that works and a process the business relies on.



Like a ribbon cutting: *the project plan ends when the ribbon is cut, which is exactly when the questions of watching, correcting and stopping it begin.*

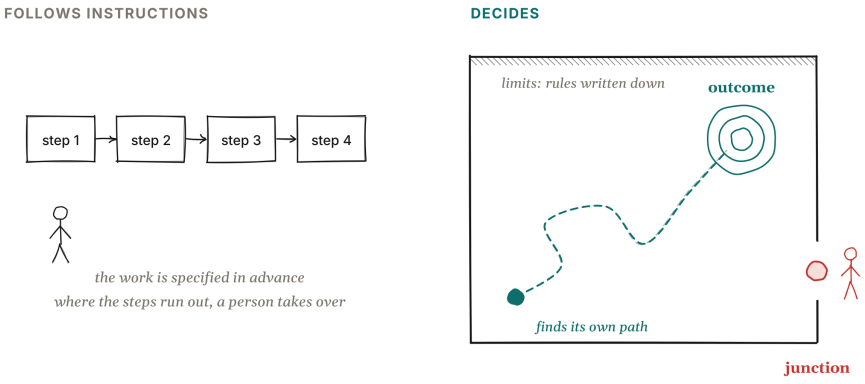
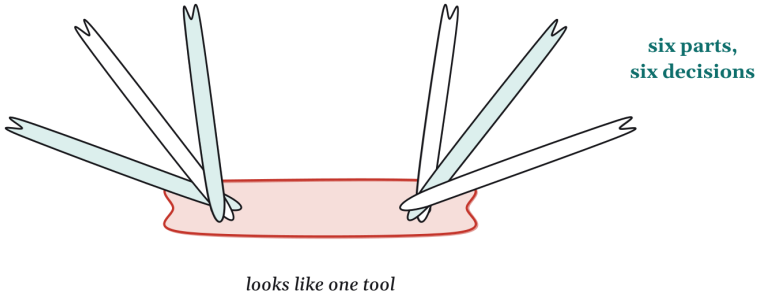


FIGURE 2 Two executors. An instruction-follower walks the written steps. An executor that decides is given an outcome, rules and limits, finds its own path, and stops at a junction where a person decides.

III

What the Executor Is Made Of

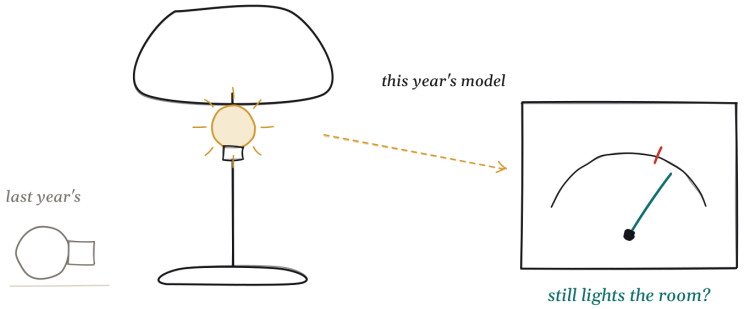
The executor that decides is often described as "the model", as if one piece of software did everything. In practice it is an assembly of six parts. A business reader does not need to build any of them. But it helps to know what they are, because each one maps to a decision the lifecycle asks the business to make.



Like a pocket knife: *it looks like one tool, but open it and there are six parts, each asking the owner to decide what it may do.*

The model

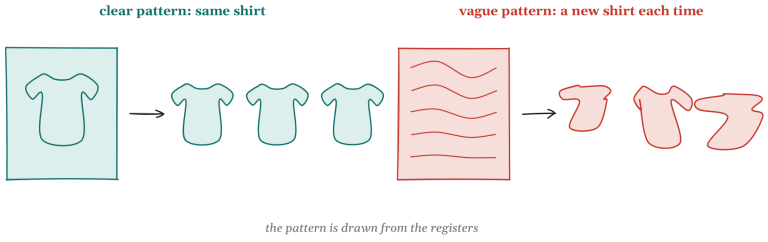
At the centre is a language model: a system trained on very large amounts of text that can read, reason about and write language. It is what lets the executor understand an invoice it has never seen, or apply a rule written in plain words. Companies rarely build their own. They buy access from a provider or run an openly available model on their own infrastructure. Models improve quickly and are replaced often. That is one reason the lifecycle grades outcomes rather than trusting a particular model: when the model changes, the grading shows whether the loop still meets its standard.



Like a bulb in a lamp: *you buy it, swap it often, and trust a light meter, not the box, to tell you the room is still lit.*

The instructions

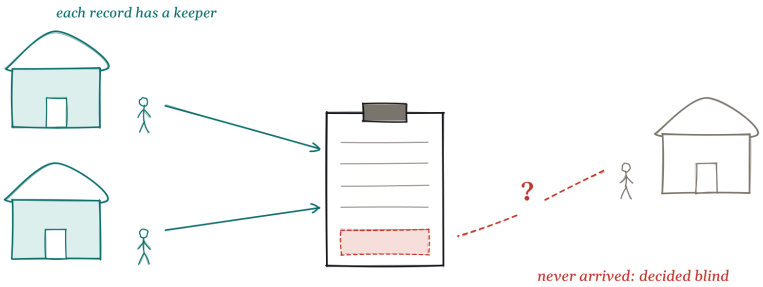
The model is told what its job is: the outcome, the rules, the limits, and what to do when a case falls outside them. In a well-designed loop, these instructions come straight from the documents the lifecycle produces. The rule register, the decision register and the junction register become the executor's brief. A loop whose rules were never gathered can only give its executor vague instructions, and vague instructions produce inconsistent decisions.



Like a sewing pattern: *a clear one, traced from agreed measurements, gives the same shirt every time; a vague one gives a different shirt from every cut.*

The context

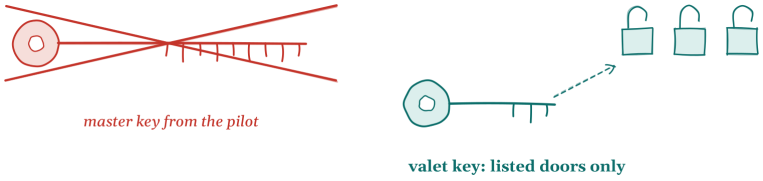
To act well, the executor must read the information the case needs: the order, the receipt, the contract, the customer's history. That information lives in the company's systems. The executor reaches it through connections that someone has to build, authorise and keep working. The context node of the canvas lists each source and its keeper. A source the executor cannot reach is a decision it will make blind.



Like a doctor's patient file: *each record is fetched from the clinic that keeps it, and a record that never arrives means a diagnosis made blind.*

The actions

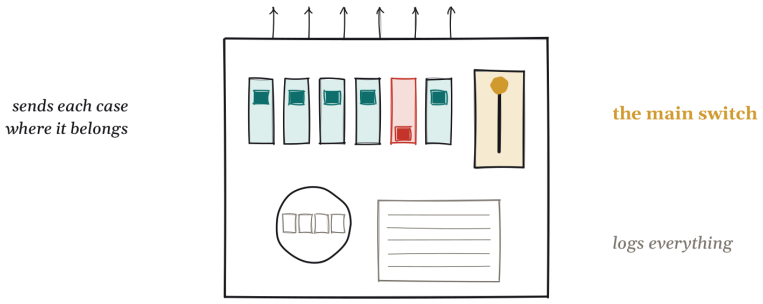
The executor changes things through tools: posting an invoice, updating a record, sending a message. Each tool is a permission granted in a particular system. This is where the business has the most control and the most exposure. The actions node lists each one with a single question: can it be undone? Permissions should match that list exactly. Permissions granted broadly for a pilot are the commonest source of executors doing things nobody intended.



Like a valet key instead of the master key: *it opens only the doors on the list, because broad keys handed out for a trial cause surprises.*

The platform

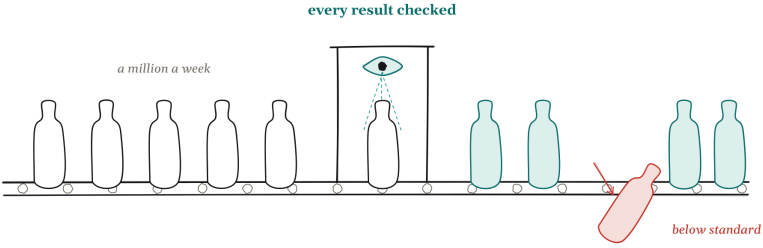
Around these parts sits the platform that runs the loop. It routes each case, calls the model, applies the limits, sends consequential cases to the junctions and exceptions to the queue, and records everything that happened. It also holds the switch that stops the loop. Some companies build this layer and many buy it. Either way, the business should insist on three things from it: a complete log of every case, a way for the named person to switch the loop off, and access for the grader to every result.



Like a building's fuse box: *it routes power to every room, trips at a limit, keeps the meter running, and has one main switch someone can pull.*

The grader

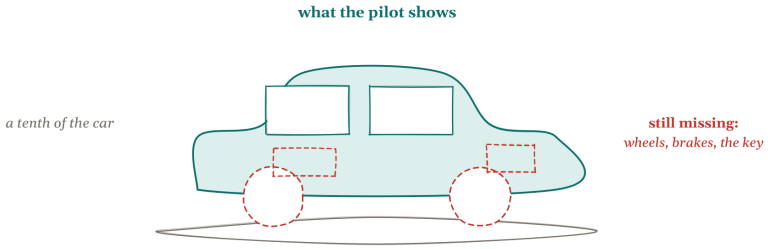
The last part watches the others. It checks every outcome against the standard, using rules, a second model trained to judge, and a small human audit. The grader is how anyone knows the loop is working. It is also the part most often left out of pilots, because a pilot can be judged by eye on a few dozen cases and a running loop cannot.



Like the inspection gate on a bottling line: *every bottle passes the sensor and is checked against the standard, because nobody can eyeball a million bottles.*

Where the pilot fits

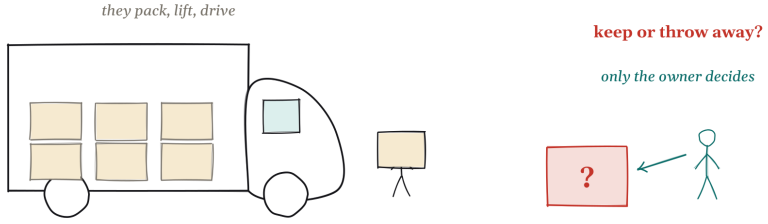
A pilot usually builds the first two parts well. It shows a capable model following good instructions on a prepared set of cases. Sometimes it connects a little context. It rarely touches real actions, almost never includes the platform's switch or a full log, and seldom has a grader. Measured by effort, a pilot is often about a tenth of what a loop running in the core requires. That is not a criticism of pilots. It is a description of the middle. Most of the remaining work is the business deciding what the other parts should allow, and the lifecycle is the order in which those decisions are made.



Like a car body on a showroom stand: *it gleams, yet the wheels, brakes and the key that switches it off are most of the work.*

Build or buy

Every part can be built, bought or rented, and the choice changes over time. The lifecycle does not depend on it. Whoever supplies the executor, the business still has to sign the rules, sort the decisions, place the junctions, write the standard and measure the economics. A vendor can run the platform. It cannot decide which disputed invoices the company is willing to pay.



Like hiring a removal firm: *they can pack, lift and drive, but only the owner can decide which boxes are kept and which are thrown away.*

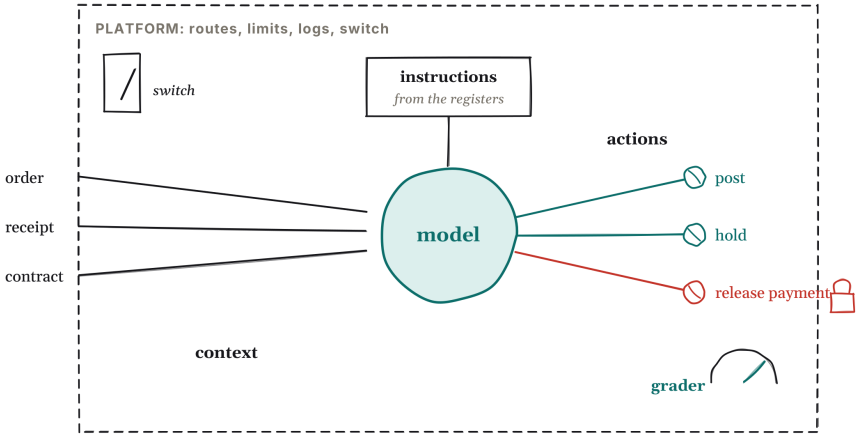
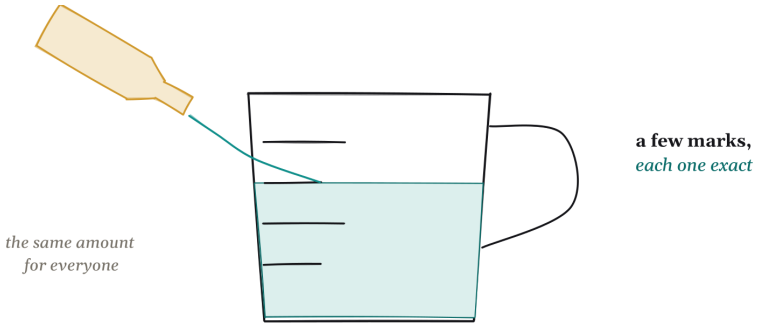


FIGURE 3 What the executor is made of. A model at the centre, briefed by instructions drawn from the signed registers, reading context through connections, acting through permissioned tools, run inside a platform that logs every case and can be switched off, with a grader on every outcome.

IV

The Words This Book Uses

A lifecycle is only as clear as its terms. This chapter defines the handful the rest of the book depends on. Each is introduced in plain words first and named second. None of them is exotic, but each has a precise meaning here.



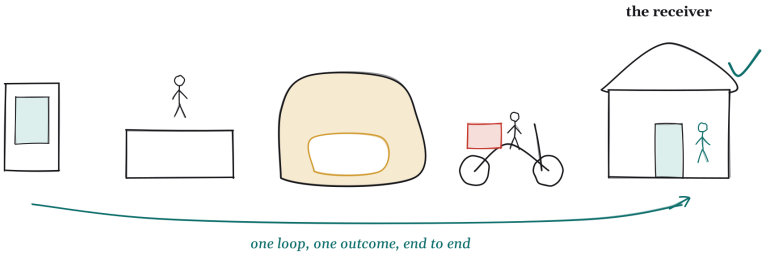
Like the marks on a measuring jug: *a handful of lines, each placed exactly, mean that everyone who pours is talking about the same amount.*

The work loop

A company does many things, but its work can be divided into outcomes that someone receives. An approved invoice ready for payment. A support ticket resolved. A new employee ready to work on day one. Each outcome has a beginning, a set of rules, decisions along the way, and a receiver who can tell an accepted result from a rejected one.

We call the unit that produces one such outcome a **work loop**. A loop owns its outcome end to end, with everything needed to produce it well. That means the rules it follows, the context it reads and the actions it may take. It also means the decisions it contains, the points where a person decides, the standard it is graded against, and the person who answers for it. A loop is smaller than a department and larger than a task. A finance function might contain a dozen loops. A single loop might cross four teams.

The loop is the unit the lifecycle moves through. You do not embed AI into "finance". You embed it into the supplier invoice loop, then the payment run, then the next loop the map points to.



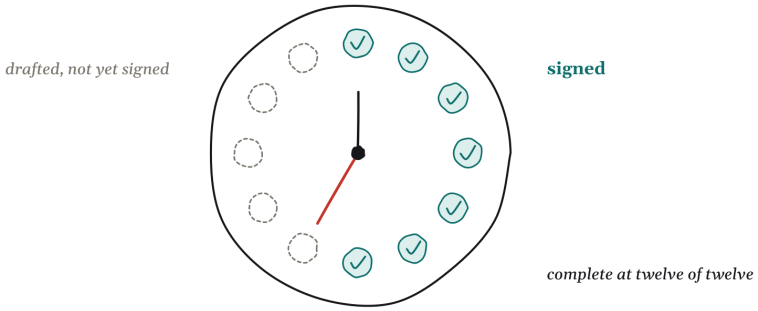
Like a pizza order from phone call to doorstep: *one outcome a customer receives, crossing the counter, the oven and the rider, owned from end to end.*

The canvas and the map

A loop is described on one page, the Work Loop Canvas. It has twelve connected parts, called nodes, filled in a fixed order. The outcome. The compensations to strip. The rules, the context, the actions. The decisions, the junctions, the exceptions, the handoffs. The standard, the people, and the economics. A node is drafted first and signed later. The canvas is complete when all twelve are signed.

All the loops of a company, placed by their value to the core of the business and by their readiness, form the Work Loop Map. The map shows where to start, where to map before building, and which loops hand work to one another and so can only scale together.

This book refers to the canvas and the map often, but does not depend on knowing them in detail. Where a step fills part of the canvas, the chapter explains what that part is.



Like a clock face: *twelve parts read in a fixed order round the dial, each drafted first and signed later, complete only when all twelve are signed.*

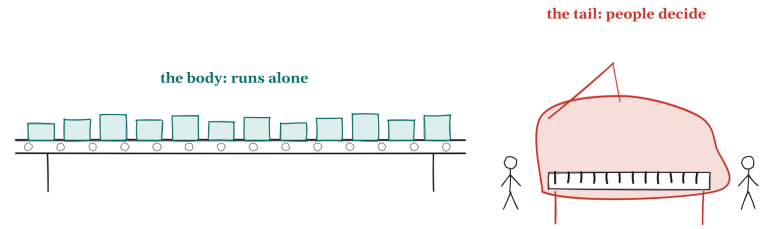
The distribution of cases

Every process handles a stream of cases: invoices, claims, tickets, orders. The cases are never all alike. How they differ is the single most useful fact about a process, and the one most often ignored.

In some processes most cases are similar and their values cluster around an average. Insurance premiums for a standard car policy, or the weekly hours on a timesheet, behave like this. Statisticians would call the shape roughly normal. In these processes, averages mean something and rules work well.

In other processes a small share of cases carries most of the value or the risk. A few invoices make up most of the spend. A few claims make up most of the payouts. A few customers make up most of the revenue. The shape has a long tail, and the usual name for it is a power law. In these processes the average misleads. The design has to treat the large body of routine cases and the thin tail of consequential ones differently.

Most real processes are mixed: a routine body and a consequential tail. **The body is where an executor runs alone; the tail is where people hold the decisions.** Reading the distribution is therefore the first act of the lifecycle, and it comes back at the end.



Like a delivery depot: *thousands of small parcels ride the conveyor on their own, while the rare grand piano needs a crew deciding every move.*

The four classes of decision

Every decision inside a loop falls into one of four classes.

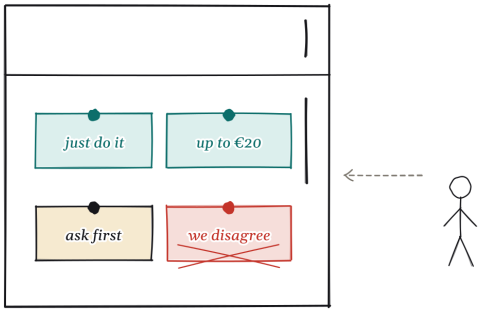
A mechanical decision follows a rule completely. Does the invoice total match the order within tolerance? The executor can take it alone.

A bounded decision needs judgement, but inside a written limit. Approve a price difference of up to 2% on an order under €5,000. The executor can take it alone, within the limit, against a standard.

A consequential decision is one where the cost of error is high or the action cannot be undone. Release a payment of €400,000. A person always decides, with the executor preparing the evidence.

A contested decision is one the organisation has never actually agreed on. Two managers would answer it differently, and both would cite policy. No design can resolve it. Someone with authority has to decide it and write it down.

Sorting every decision into these classes is one of the central acts of the lifecycle. Contested decisions are the most important to find, because they are the ones that surface as incidents if nobody finds them first.

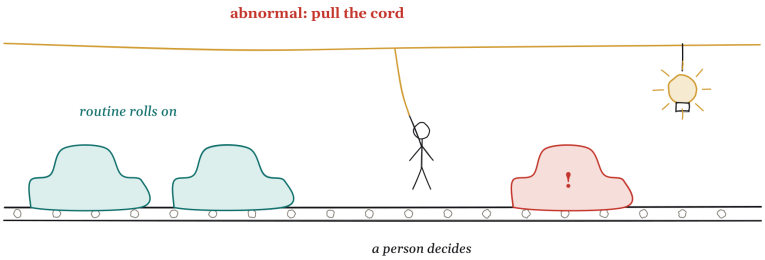


Like house rules on a fridge: *some things a teenager just does, some within a limit, some only after asking, and some the parents never agreed on.*

The junction

A **junction** is a point where a person decides instead of the executor. It is defined by three things: who decides, what evidence they see, and how the action is reversed if it was wrong. A junction is placed by the reversibility and the cost of the action, not by how comfortable anyone feels. An action that can be undone cheaply rarely needs a junction. An action that cannot be undone almost always does.

The ancestor of the junction is the cord on a Toyota assembly line that any worker could pull to stop the line. The routine runs on its own. An abnormal case stops it. A person decides. Part of a junction's design is the plain question of who can stop the loop altogether.



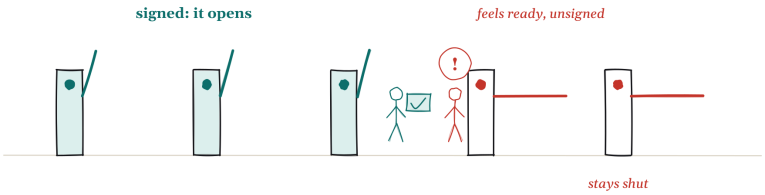
Like the cord on Toyota's assembly line: *routine cars roll on by themselves, an abnormal one stops the line, and a person decides what happens next.*

The gates

The lifecycle moves a loop through five gates, called the **Scale Gates**. Each gate opens only when particular nodes of the canvas are signed.

At G0, Mapped, every node has been drafted. At G1, Shadow, the loop may run beside the current process without touching anything. At G2, Bounded, it may act alone inside written limits. At G3, Owned, named people answer for it and its economics have been measured. At G4, Core, all twelve nodes are signed and it runs as part of the business.

Scale follows signatures, not confidence. A loop that feels ready but has unsigned nodes is not ready.



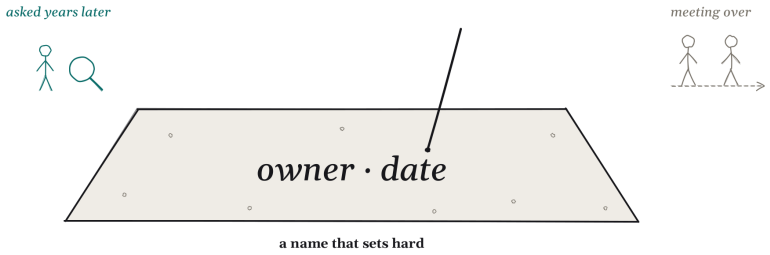
Like a row of ticket barriers: *each one opens only for a stamped ticket, so a loop moves forward on signatures, however ready it feels.*

The signature

The last term is the simplest and the one the lifecycle leans on most. A **signature** in this book means that a named person has read a document, accepted it as true or as decided, and put their name and the date to it. It does not need ink. It needs a name, a date, and a record that survives the meeting.

Signatures matter because they turn agreement into accountability. A meeting can agree that the tolerance is 2%. A signed rule register says who decided it,

when, and where it is written. When an auditor, a regulator or a new manager asks why the loop behaves as it does, the signatures are the answer.



Like initials pressed into wet cement: *a name and a date that set hard and stay long after the meeting, ready for whoever asks why later.*

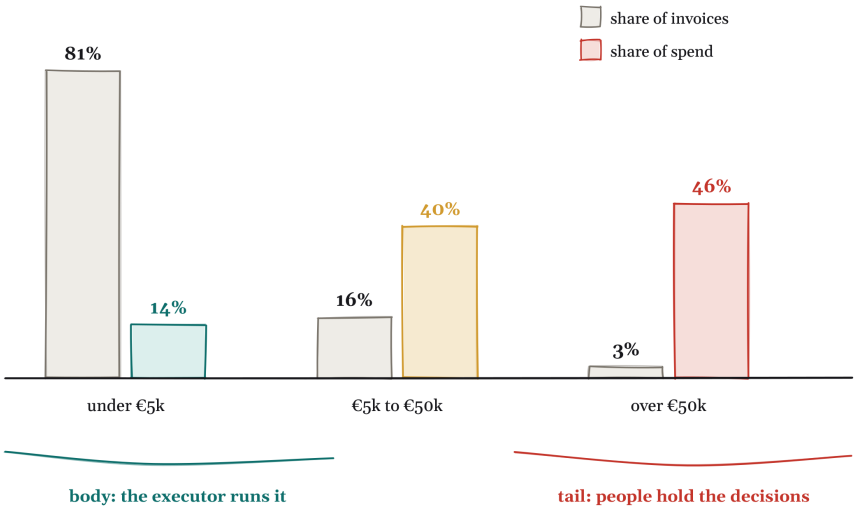


FIGURE 4 Body and tail, as read at the manufacturer. Most invoices are small and carry little of the spend; a few large ones carry almost half. The executor runs the body; people hold the tail.

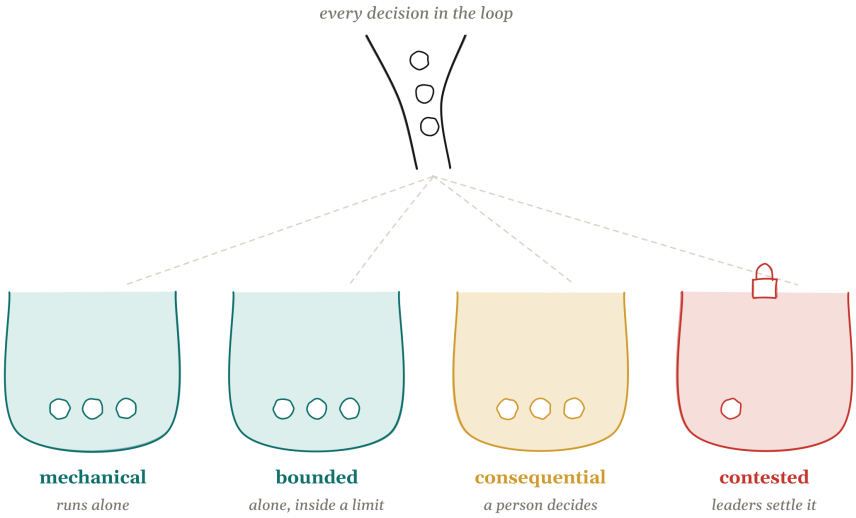
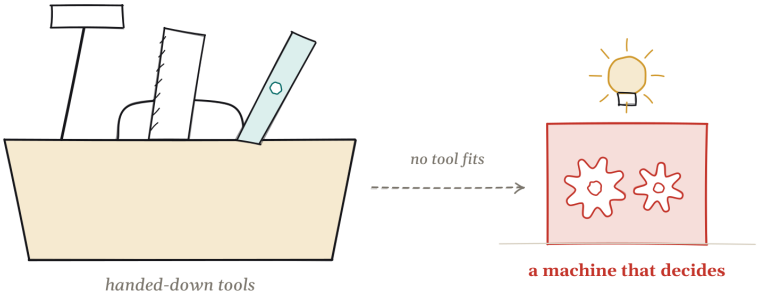


FIGURE 5 The four classes of decision. Every decision in a loop is sorted into one of four bins. The contested bin stays locked until a leader with authority settles each decision in writing.

V

Lifecycles Before This One

The idea of a lifecycle is old. Organisations have long broken large changes into stages, with a review between each. Several of those lifecycles shaped how companies work today. Each solved a real problem, and each carries something this lifecycle keeps. None was built for an executor that decides.

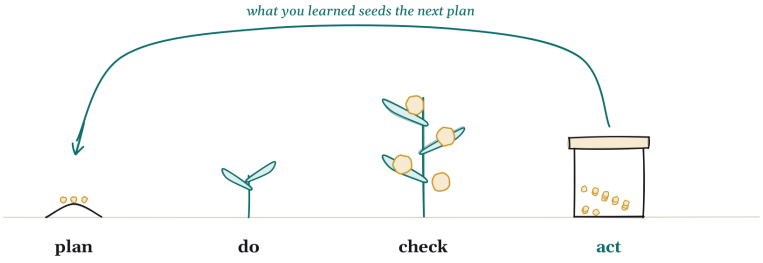


Like a toolbox handed down for generations: *every old tool still earns its place, but none was made for the machine now on the bench.*

The plan-do-check-act cycle

In the 1930s, the statistician Walter Shewhart described production as a cycle of specifying, producing and inspecting. W. Edwards Deming carried the idea to Japanese industry in the 1950s, and it became known as plan, do, check, act. Its lasting contribution is the loop itself: work is never finished, and what you learn from checking feeds the next plan.

This lifecycle keeps that shape. Its last step returns to its first.

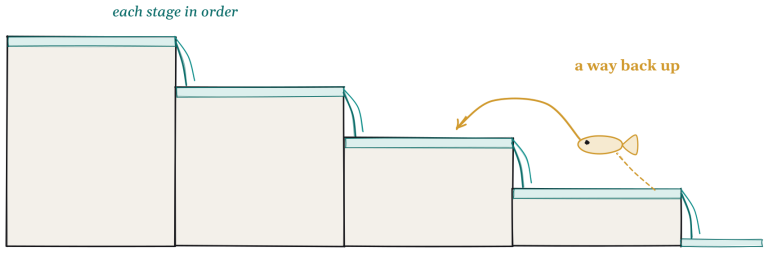


Like a gardener who saves seeds: *what this year's harvest teaches, and its best seeds, go straight back into next spring's planting.*

The waterfall

In 1970, Winston Royce published a paper on managing large software projects. It is remembered for a diagram of stages cascading from requirements to operations, each completed before the next began. The diagram became the "waterfall" and dominated large projects for decades. Royce himself warned that the pure sequence was risky and recommended building a first version to learn from. That warning was mostly ignored.

The waterfall's lasting idea is that some things must be settled before others. Its failure was treating every stage as complete before the next began, with no way back. This lifecycle keeps the order but expects to loop back when a later step uncovers something an earlier one missed.

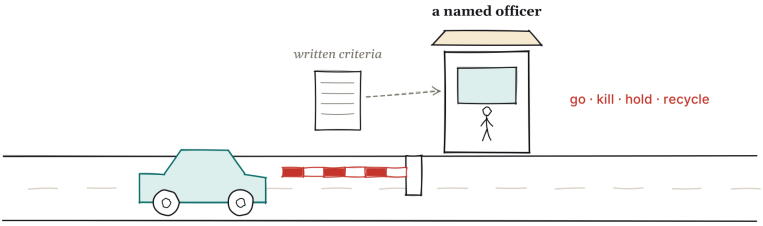


Like a waterfall with a fish ladder beside it: *the water still falls step by step, but a fish can climb back when a lower pool finds something missed.*

Stage-gate reviews

In 1990, Robert Cooper described the stage-gate process for new products. Work moves through stages, and between each stage sits a gate where named reviewers decide go, kill, hold or recycle against written criteria. The gates protected companies from spending heavily on products that should have been stopped early.

This lifecycle takes its gates from that idea. Each step ends with a gate, and each gate has a named signer and written criteria.

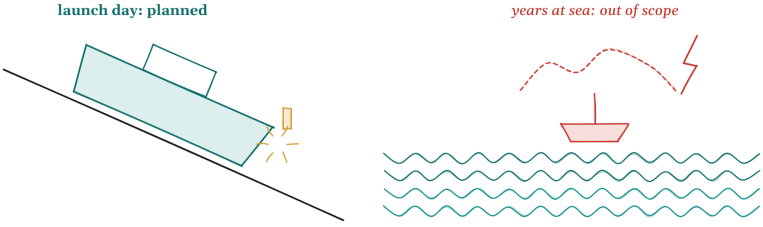


Like a border post: *at each crossing a named officer checks the papers against written rules, then lifts the barrier, holds the car, or sends it back.*

The data-mining cycle

In 1999, a consortium of companies published CRISP-DM, the cross-industry standard process for data mining. It ran from business understanding through data understanding, preparation, modelling and evaluation to deployment, with arrows looping back between phases. It was the first widely used lifecycle for building models, and it put business understanding first.

Its blind spot was the end. Deployment was a single phase, often a report. What happened when a model ran inside a process for years, with its inputs drifting and its owners changing, was mostly out of scope.



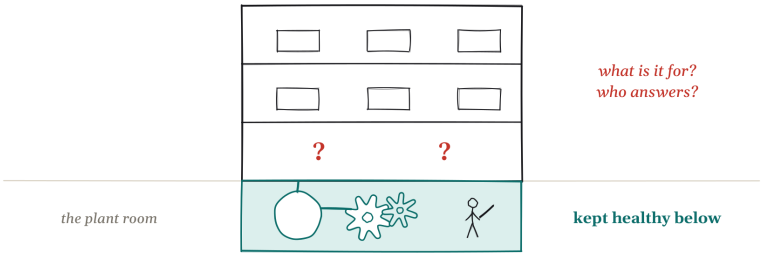
Like a ship launch: *the design and the build are planned with care, the bottle breaks, and the years at sea afterwards are nobody's plan.*

Service management and MLOps

Two later traditions addressed running systems over time. Service management, codified in the British government's ITIL guidance from 1989, organised how IT services are designed, handed over, operated and improved. Machine-learning operations, which grew after a 2015 paper on the hidden technical debt in machine-learning systems, applied similar discipline to models: monitoring, retraining, versioning.

Both are essential, and both sit mostly below the business. They tell you how to keep a system healthy. They do not tell you which decisions it should be

allowed to make, who answers for them, or whether the process it serves was worth automating in the first place.

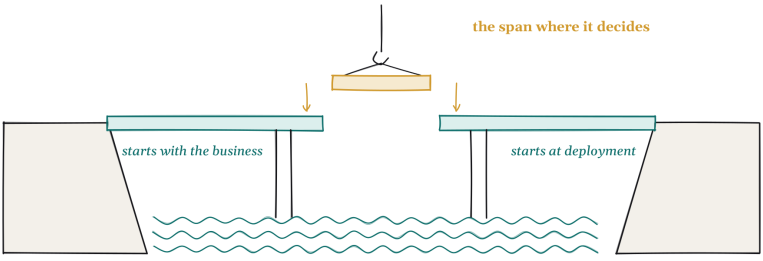


Like a building's plant room: *engineers keep the boiler and pumps healthy below, but what the floors above are for is decided somewhere else.*

What was missing

Put these lifecycles side by side and a gap appears. Some begin with the business but end at deployment. Others begin at deployment but assume the business questions were settled. None was designed for an executor that decides inside a core process, where the business questions and the operating questions cannot be separated.

This lifecycle tries to close that gap. It keeps the loop from plan-do-check-act and the order from the waterfall. It keeps the gates from stage-gate reviews and puts business understanding first, as CRISP-DM did. And it keeps the long view of service management. What it adds is a sequence of business decisions, each signed, that has to be completed before an executor that decides may act alone.



Like a bridge begun from both banks: *one crew starts from the business, one from deployment, and the middle span where the executor decides was never laid.*

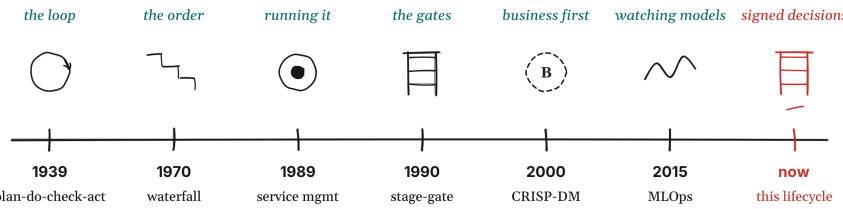


FIGURE 6 Lifecycles before this one. Each earlier lifecycle left one idea this lifecycle keeps; none was built for an executor that decides.

VI

The Lifecycle at a Glance

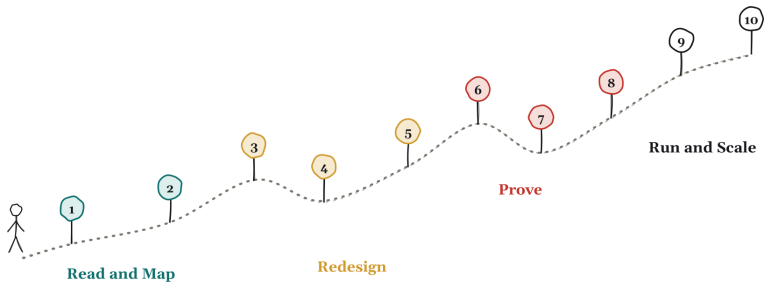
The lifecycle has ten steps in four phases. Before taking them one by one, it helps to see the whole.

The first phase, Read and Map, establishes what is true today. Step 1, Read, looks across the company's loops and asks where the value sits and where to start. Step 2, Map, takes one loop and records how its work really runs, with a measured baseline of time, cost and quality.

The second phase, Redesign, decides what the work should become. Step 3, Strip, removes the steps that exist only because of constraints that no longer apply. Step 4, Sort, gathers the rules and sorts every decision into its class, and it puts the contested ones in front of someone who can settle them. Step 5, Design, draws the loop: the junctions, the exception queue, the handoffs and the standard it will be graded against.

The third phase, Prove, earns trust in stages. Step 6, Shadow, runs the loop beside the current process without touching anything, and compares results. Step 7, Bound, lets it act alone on the routine body of cases, inside written limits. Step 8, Own, puts named people in place and measures the economics against the baseline.

The fourth phase, Run and Scale, makes the loop part of the business and moves on. Step 9, Core, runs it in production with drift checks, a monthly review and a tested way to switch it off. Step 10, Scale, reviews the whole map again, decides which loop comes next, and plans how people's roles change. Its output is the input to the next Read.



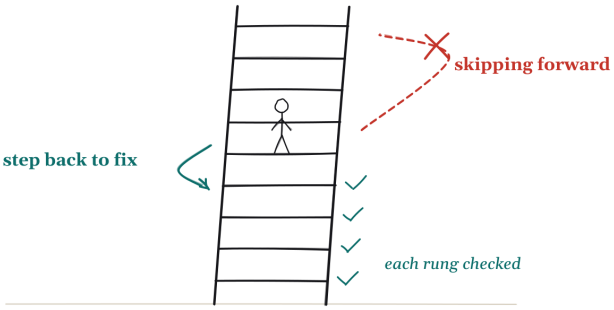
Like a trail map at the trailhead: *ten waymarks in four stretches, seen whole before the walker takes the first step.*

Three rules that hold across all ten steps

Each step answers one question, and the answer is written down. A step that ends with a discussion but no document has not ended.

Each step ends at a gate with a named signer. The signer is the person whose authority the answer needs. That is usually the process or loop owner, sometimes a control owner, finance or the executive sponsor. The gate either opens or it does not.

The steps run in order, but the lifecycle can loop back. If shadow running reveals a rule nobody recorded, the work returns to Sort for that rule, and the rule register is signed again. Looping back is normal. Skipping forward is the failure.



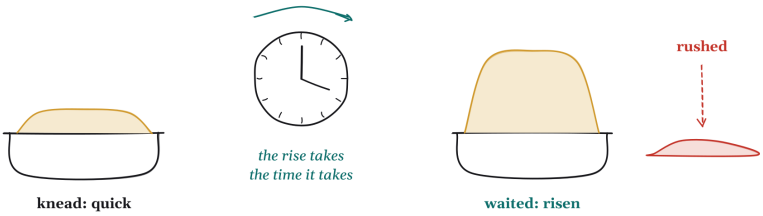
Like climbing a ladder: *each rung is checked as you go, stepping back down to fix one is normal, and leaping rungs is how people fall.*

How long it takes

For a single loop of moderate size, the first six steps usually take between four and six months. Read takes about two weeks, Map three, Strip one, Sort three, Design two, and Shadow six. Bound and Own add two to three months, mostly measurement. A loop typically reaches Core between seven and ten months after Read begins.

That can sound slow next to a pilot built in three weeks. But the pilot answered one question, whether a model can do the task. The lifecycle answers the forty or so questions a business needs answered before it relies on the work. Most of the elapsed time is waiting for evidence: six weeks of shadow running, eight weeks of measured economics. That time cannot be compressed without guessing, and guessing is what stalls programmes in the middle.

The second loop is faster than the first. The map already exists, the people know the steps, and the templates have been used once. By the third or fourth loop, a company is usually running several lifecycles at once, each at a different step. Part Six describes how.

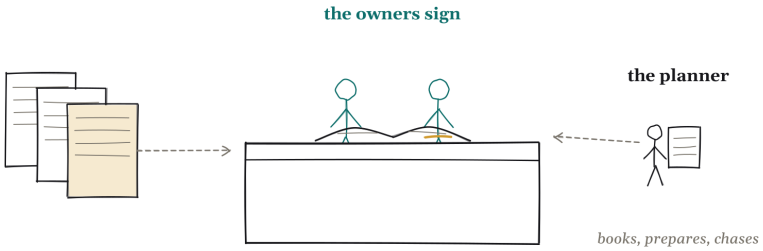


Like bread dough proving: *kneading is quick, but the rise takes the hours it takes, and a loaf rushed into the oven collapses.*

Who runs it

Every lifecycle needs a person who carries it from step to step: who books the sessions, prepares the documents, chases the signatures and knows what each gate needs. In this book that person is called the business engineer. The title matters less than the job. It can be held by someone in finance, operations, transformation or IT. It must be held by someone who understands the business well enough to ask the right questions, and the technology well enough to know what the executor can and cannot do.

The business engineer does not sign the gates. Owners and control holders sign. The business engineer makes sure there is something worth signing.



Like a wedding planner: *books the room, prepares the papers and chases everyone, but never signs the register; the people who own the promise do.*

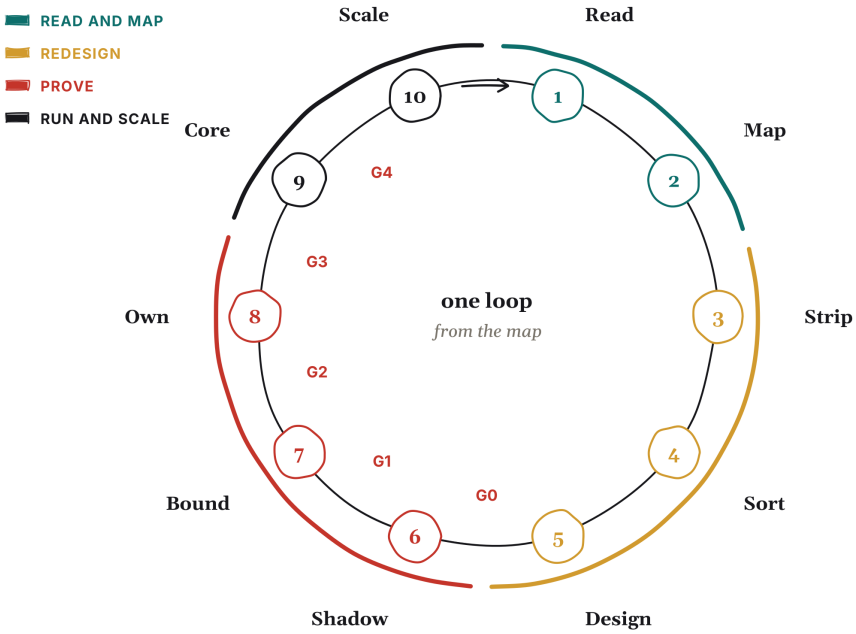
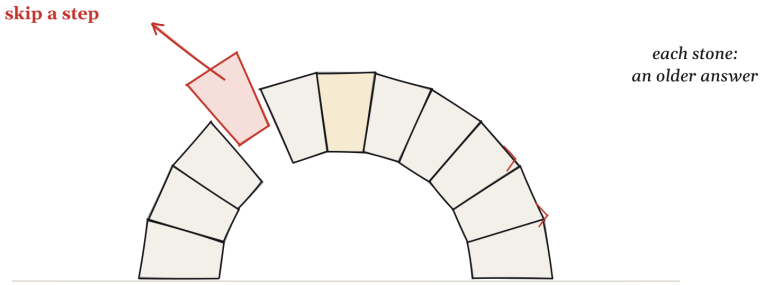


FIGURE 7 The lifecycle at a glance. Ten steps in four phases, with the Scale Gates along the way. Step 10 feeds the next Read, so the lifecycle closes into a loop.

VII

Why the Lifecycle Has This Shape

The ten steps are not an arbitrary checklist. Each one answers a question that older fields of knowledge have already worked on. Knowing those answers makes the steps easier to run, and easier to defend when someone wants to skip one.

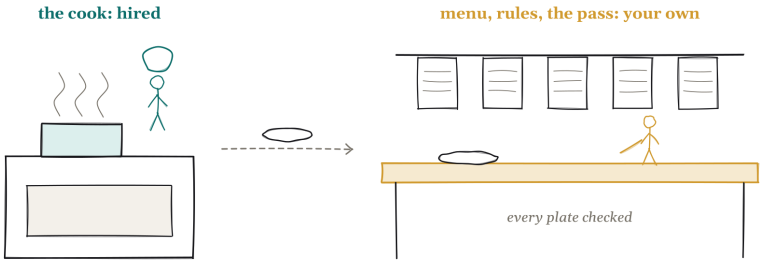


Like a stone arch: *every stone rests on an answer older fields already proved, and pulling one out to save time brings the arch down.*

Two kinds of machinery

For most of the history of computing, there were two ideas of how to make a machine reason. One wrote the reasoning down as explicit rules, in the tradition of George Boole's laws of thought. Rules can be read, checked and audited, but they break at the edges, where the world is richer than anyone wrote down. The other let machines learn from examples. Learned models handle variety well, but they come with no guarantees, and they are strong and weak in ways nobody can see in advance.

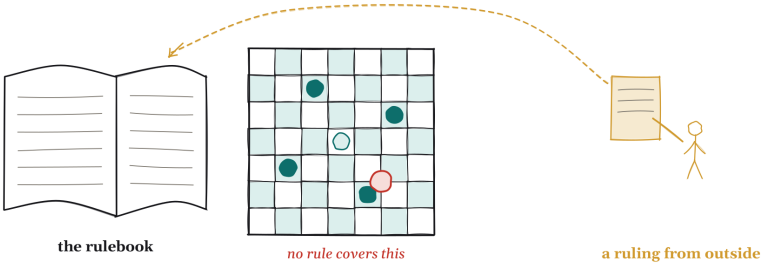
Researchers now combine the two, and call the result neurosymbolic AI. An embedded loop is such a combination. The model is the learned half, and a company can buy it. The written half is the rule register, the decision register, the limits, the junctions and the standard. A company cannot buy that half, because it describes its own work. Steps 2 to 5 build it. Steps 6 to 9 test it and keep it true.



Like a restaurant kitchen: *you can hire a skilled cook, but the menu, the house rules and the pass where each plate is checked must be your own.*

Rules never cover everything

In 1931, the logician Kurt Gödel showed that any rule system powerful enough to be useful leaves some truths it cannot reach from inside. Businesses meet a practical version of that limit in three forms: the tail of rare cases, the tacit knowledge in experienced people's heads, and the decisions the company never agreed on. The lifecycle does not pretend these away. Sort puts contested decisions in front of someone with authority, because only a person outside the rule system can extend it. Design sends the tail to people at junctions. Shadow surfaces the tacit rules that people apply without having written them down.

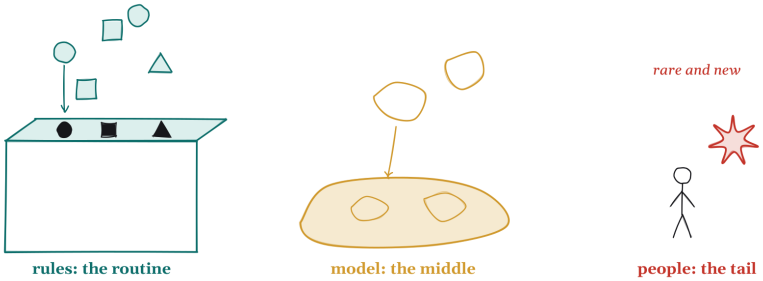


Like a rulebook at a game table: *some positions arise that no rule covers, and play waits until a referee outside the book writes a new ruling.*

Only variety absorbs variety

The cybernetician W. Ross Ashby stated a law that every designer of control meets: a regulator must have at least as much variety as the disturbances it faces. A loop that meets a hundred kinds of case cannot be run by a rule set that knows ten. The lifecycle spreads the variety across three things. Written rules take the routine body, where cases repeat. The model adds variety in the middle, where cases differ in ways rules would struggle to list. People hold the

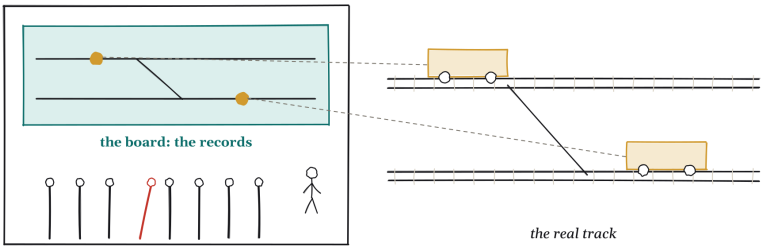
tail, where cases are rare, consequential and new. Design is largely the work of making that allocation explicit.



Like a child's shape sorter: *ten holes cannot take a hundred shapes, so common ones drop through, odd ones press into soft clay, and rare ones go to a hand.*

The record set is the regulator

Ashby's colleague Roger Conant proved with him a second result: every good regulator of a system must be a model of that system. A company that wants to control an embedded loop must hold a model of it. The map, the baseline, the registers, the standard, the shadow report and the charter are that model. They are not paperwork produced for an auditor. They are what lets the company see what the loop is doing and correct it. That is why every step ends in a signed document.

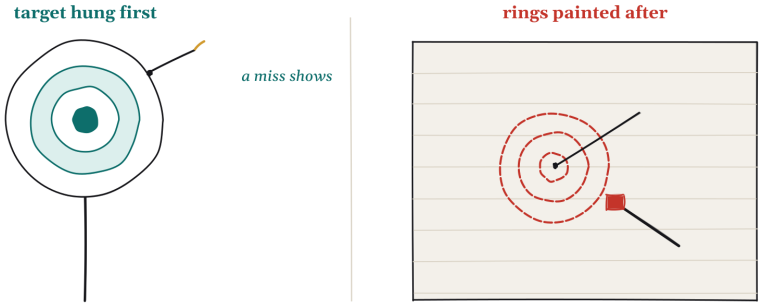


Like a railway signal box: *the signaller steers trains they cannot see by a board that mirrors the track, so the board is the control, not paperwork.*

Claims that can fail

The philosopher Karl Popper argued that a claim worth making is one that a result could prove wrong. Shadow and Bound are built that way. The agreement the loop must reach, body and tail apart, is written down before the cases arrive. The limits are written before the loop acts. If the evidence falls

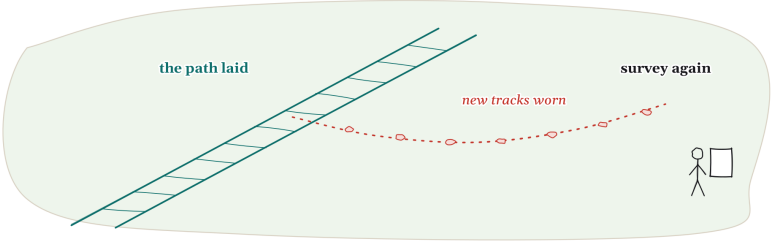
short, the gate does not open. A lifecycle that measures first and sets the bar afterwards can always declare success, and so it learns nothing.



Like an archer who paints the rings after the arrow lands: *always a bullseye, nothing learned. Hang the target first, and a miss can show.*

The loop changes what it reads

Norbert Wiener described control as a closed loop: act, sense the result, compare it with the goal, adjust. Core is the loop running. But a running loop also changes the distribution it was designed for. Suppliers adapt, volumes shift, the tail grows or shrinks. That is why the last step re-reads the distribution and returns to the map. **The lifecycle is itself a closed loop: its output is the next reading.**



Like paving a footpath across a park: *once it is laid, people walk differently and wear new tracks, so the gardener surveys the lawn again.*

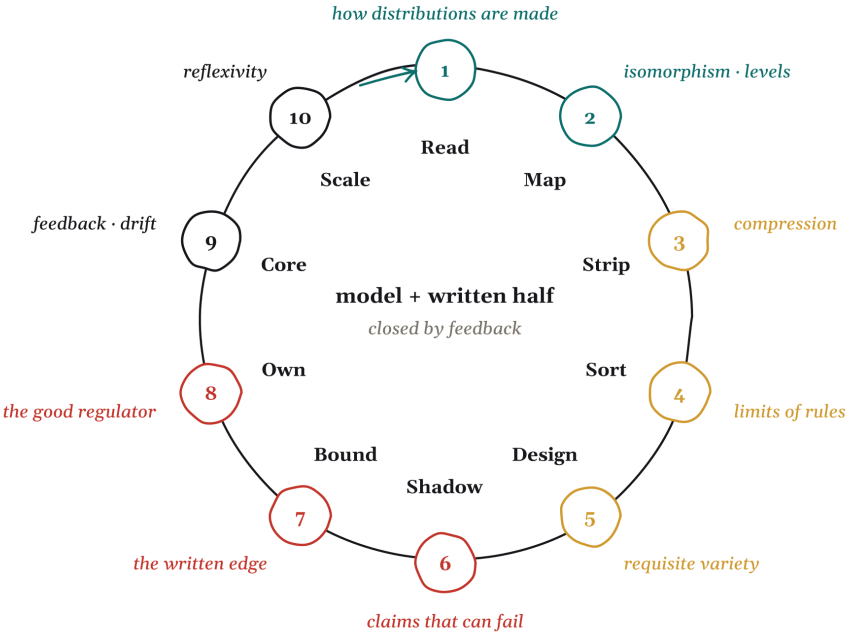
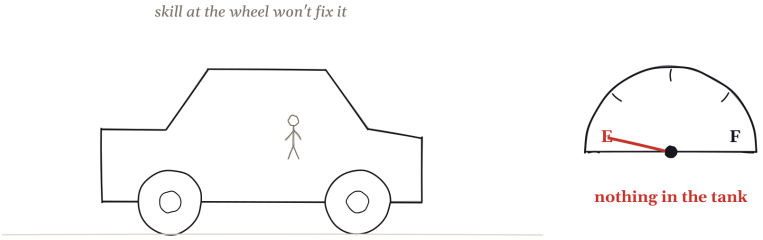


FIGURE 8 Why the lifecycle has this shape. Each step answers a question older fields already worked on: reading distributions, mirroring the work, compression, the limits of rules, requisite variety, falsifiable tests, the good regulator, feedback and reflexivity.

VIII

Before Step One

The lifecycle assumes a few things are in place before the first reading. Most stalled programmes lacked at least one of them, and no amount of work at later steps makes up for it.



Like a car with an empty tank: *however skilled the driver, nothing done at the wheel later makes up for what was never put in.*

Five conditions

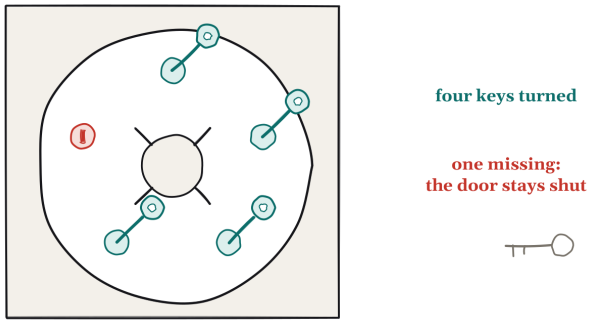
The first is a sponsor with authority over the loops in question. Embedding crosses departments, and only someone senior enough to settle a disagreement between them can keep the lifecycle moving. In most companies that means an executive committee member. A sponsor who cannot settle a contested decision will watch the programme stop at Sort.

The second is a business engineer: one named person who will carry the first loop from Read to Core. It can be a part-time role for the first loop, but it cannot be nobody. Lifecycles run by committee rarely reach Bound.

The third is time from the people who know the work. Mapping, stripping and sorting need operators, team leads and process owners in the room, for several half-days over a few weeks. If their managers will not release them, the map will be drawn by people who do not do the work.

The fourth is access. Someone must be able to pull a few months of cases for the distribution and the baseline, and later connect the executor to the systems it must read. A loop whose data cannot be reached cannot be read, let alone embedded.

The fifth is finance's agreement to measure. Finance should agree at the start that it will sign the baseline method and, later, the economics. Programmes that bring finance in at the end usually find their savings disputed.

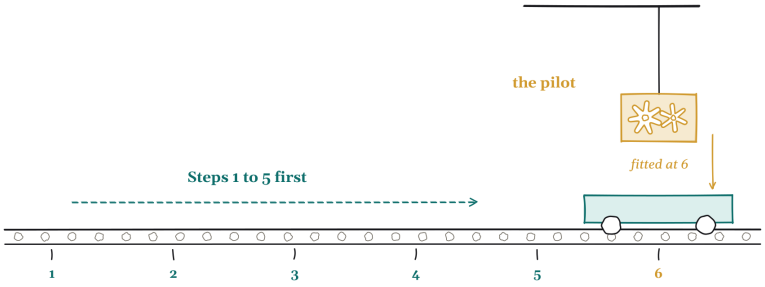


Like a vault that opens only when five keys turn: *the sponsor, the business engineer, the operators' time, data access and finance each hold one.*

If a pilot already exists

Many companies reach this book with a pilot already built. That is useful, not wasted, but it does not let the loop skip ahead. The right move is to run Steps 1 to 5 quickly around the pilot. Check on the map that this loop is the right one to start with. Map its real process and measure the baseline. Strip, sort and design. Then bring the pilot in at Step 6, where it becomes the executor in shadow. Most of what the pilot built survives. What changes is that it now runs against signed rules, inside designed junctions, and is graded against a written standard.

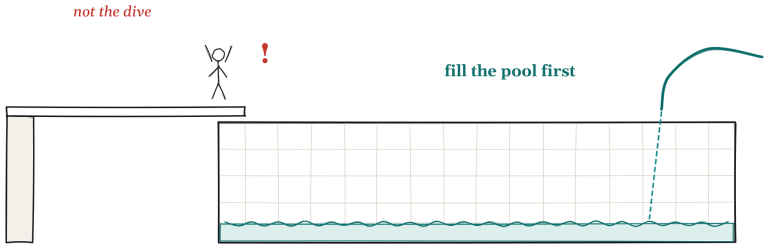
If Step 1 shows that the pilot's loop is a poor place to start, it is better to learn that in two weeks than in two quarters. The pilot can wait on the map until its loop's turn comes.



Like an engine built before the car: *keep it, run the chassis down the line first, then fit it at its station, the sixth step.*

Who should not start yet

A company that cannot meet the five conditions should start with the conditions, not the loop. That usually means one decision by the executive committee: name a sponsor, name a business engineer, free the operators' time, open the data, and bring finance in. It is a short list, and it is the difference between a programme that crosses the middle and one that stalls in it.



Like a diving board over an empty pool: *the first job is filling the pool, not perfecting the dive.*

PART TWO

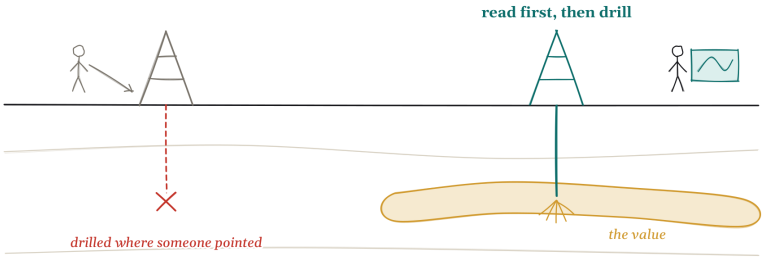
Read and Map

IX

Step 1: Read

The step at a glance	
Question	Where does value sit, and which loop do we start with?
Starts when	An executive sponsor has asked for AI to reach the core, and has given the time of the people who know the work
Signed	The Work Loop Map, with the first loop chosen
Signer	The executive sponsor
Exit gate	One loop selected, its owner named, its distribution read
Typical duration	Two weeks

The lifecycle begins with a reading, not a project. Before anyone touches a process, the company needs a picture of its loops, where value sits among them, and which one to start with. Most programmes skip this step. They start with whatever a vendor demonstrated, or whatever a keen manager volunteered. Then they discover months later that they have embedded AI into work that did not matter much.



Like a prospector reading the survey before drilling: *most wells go down wherever someone pointed and come up dry, while the reading finds where the value lies.*

The work

The reading has three parts.

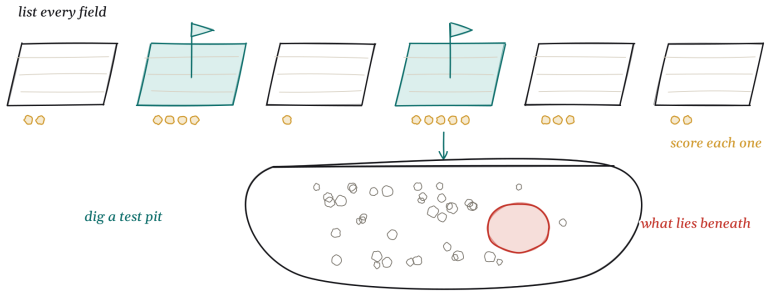
The first is listing the loops. A working session with the heads of the main functions lists the outcomes the company produces, one per line, in plain words. Approved supplier invoices. Resolved customer tickets. Qualified leads.

Closed month-end books. A mid-size company usually finds between twenty and fifty. Each is named by its outcome, not by the department that produces it, because many outcomes cross departments.

The second is placing each loop on the map. Two scores are enough for a first reading. Value to the core asks how much this loop matters to the business: its volume, its cost, its effect on customers, cash or risk. Readiness asks how close it is to being embedded: how well its rules are known, how measurable its outcome is, how much of its context is already in systems. Both are scored from one to five by the people who know the work, and the scores are argued over. The argument is useful. It is often the first time a leadership team has compared its processes side by side.

The third is reading the distribution of the candidate loops. For the two or three loops that score highest, someone pulls a few months of cases and looks at their shape. How many cases are there? How do their value and difficulty spread? Is there a routine body and a consequential tail, or are the cases all alike, or all different?

The distribution decides more than it seems to. A loop with a large routine body and a clear tail is a good first candidate, because the executor can take the body while people keep the tail. A loop where every case is different and consequential, such as a merger review, is a poor first candidate however valuable it is. A loop with high volume but trivial value may be easy, but it will not teach the company much or pay for the effort.



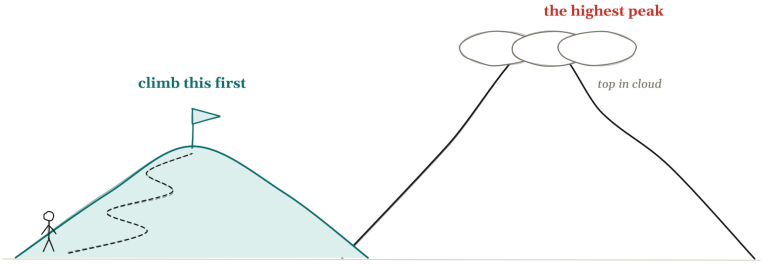
Like a farmer choosing where to plant first: *list every field, score each one, then dig test pits in the best two to see what lies beneath.*

Choosing the first loop

The first loop has a job beyond its own savings. It teaches the company the lifecycle. So it should have four properties. Its cases repeat, so evidence accumulates quickly. Feedback is fast, so the company learns within weeks

rather than quarters. Its outcome can be measured. And its owner is willing. A first loop owned by a reluctant manager will stall at Sort, where the owner has to decide things.

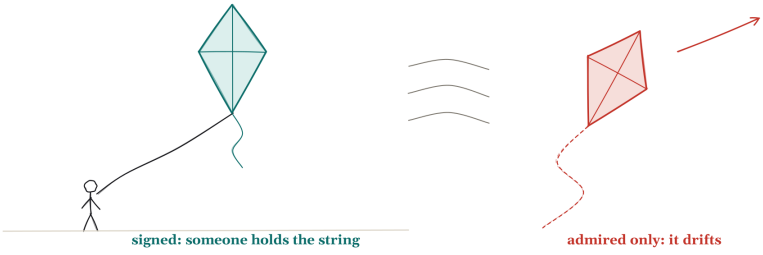
The best first loops are rarely the most valuable ones on the map. They sit in the top right quadrant, high on value and high on readiness, with a body and a tail. The most valuable loops often sit lower on readiness. Their turn comes once the company has learned the steps.



Like the first walk of a hiking holiday: *not the highest peak, but a near hill with a clear path and a visible top.*

What gets signed

The executive sponsor signs the Work Loop Map with the first loop marked. The signature commits three things: that this loop goes first, that its owner is named and has agreed, and that the people who know the work will be available for the next steps. A map that is admired but unsigned commits nothing, and the programme drifts.

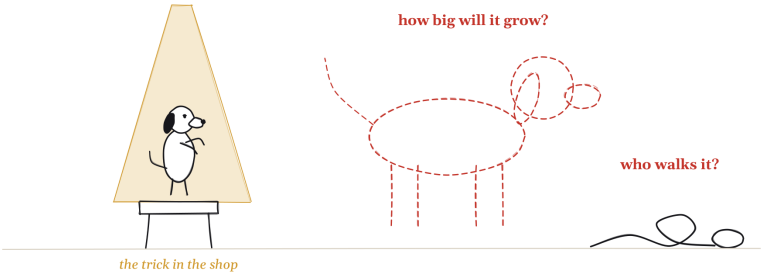


Like a kite: *a map that is only admired drifts off on the first wind, while a signed one has someone holding the string.*

How it fails

The common failure is choosing by enthusiasm instead of by the map. The loop that a vendor demonstrated, or the one a senior leader cares about, goes

first regardless of readiness. A second failure is skipping the distribution. Without it, nobody knows whether the loop has a body that an executor could run, and the design that follows treats every case the same. A third is naming an owner who has not agreed. Owners are named in a meeting they did not attend, and the loop has no one who will make its decisions.

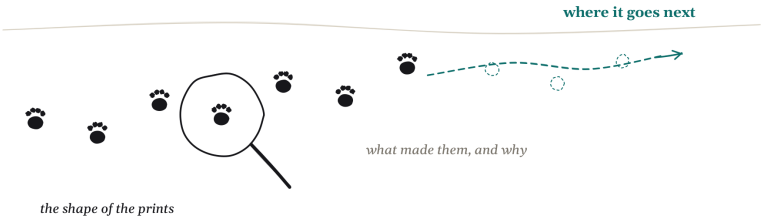


Like choosing a puppy because it did a trick in the shop: *nobody asked how big it grows, or whether anyone has agreed to walk it.*

What the foundations add

A distribution is the footprint of a mechanism. Many small independent causes produce a normal shape, where the average case is typical. Compounding, where big gets bigger, produces a heavy tail, where a few cases carry most of the value. Rules and thresholds leave their own marks, such as a pile of invoices just under an approval limit. So the reading should ask why the shape is what it is, not only what it is. A shape whose cause is understood can be predicted when the business changes. A shape that is merely observed cannot.

The reading also segments on three axes, not one. Value is the usual axis. Difficulty and risk can cut differently: a small invoice from a new supplier in a new country may belong in the tail even though its amount is trivial. Write the reading down as a prediction, such as "most invoices will stay under €5,000 next year". Later steps can test it, and Scale will.



Like a tracker reading prints in the snow: *the shape tells you what made them, and knowing that lets you predict where the trail goes next.*

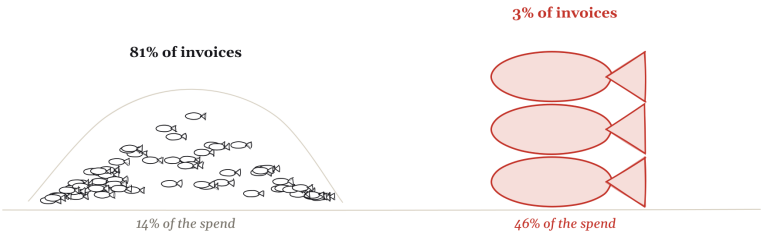
At the manufacturer

The manufacturer's chief financial officer sponsored the programme. In two half-day sessions, the heads of finance, operations, sales, service, HR and IT listed thirty-four loops across six chains of work, from quoting to collecting cash.

Placing them took longer than listing them. Order fulfilment scored highest on value, but its rules were scattered across three plants and its readiness was low. Customer quoting was valuable and contested, and nobody wanted to start there. Supplier invoice approval scored four on value and three on readiness. It was the most frequent case in the company, it touched every plant, and finance could measure it to the cent.

The finance team pulled six months of supplier invoices. There were about 7,400 a month. Their shape was the clearest finding of the reading. Eighty-one percent of invoices were under €5,000, and they carried 14% of the spend. Sixteen percent fell between €5,000 and €50,000 and carried 40%. The last 3%, each over €50,000, carried 46% of everything the company paid its suppliers.

That was a routine body and a heavy tail, which made it the right first loop. The head of accounts payable agreed to own it. The CFO signed the map with the invoice loop marked first and goods receipt marked to map next.



Like a day's catch: *the heap of small fish is most of the count but little of the weight, while three big ones weigh almost half.*

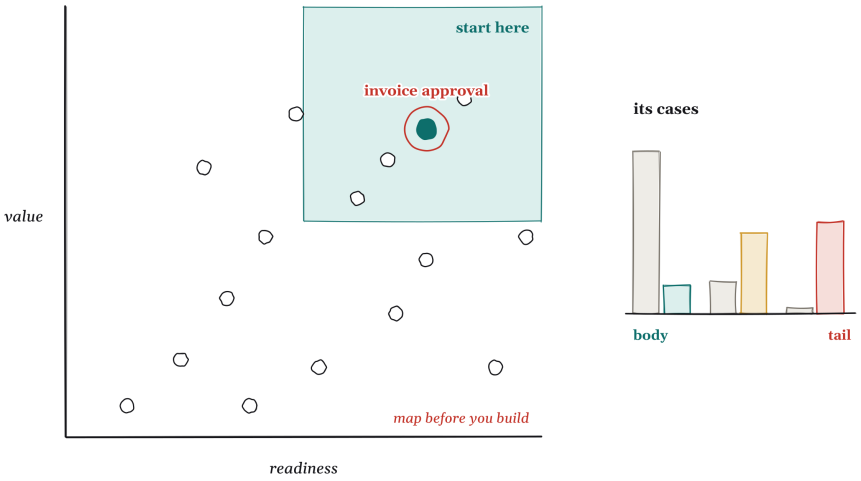


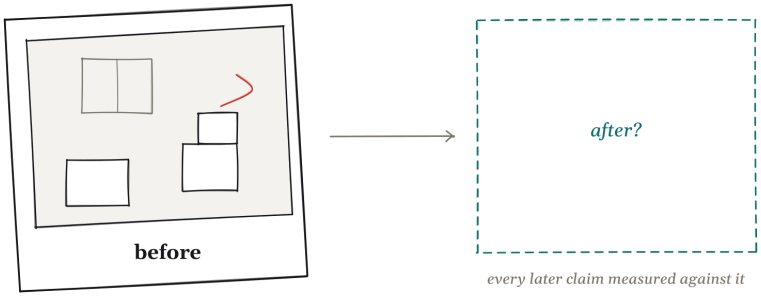
FIGURE 9 Step 1, Read. The loops placed by value to the core and readiness, with the first loop chosen in the upper right, and the shape of its cases read before anything else.

X

Step 2: Map

The step at a glance	
Question	How does the work really run today, and what does it cost?
Starts when	The loop is chosen and its owner has agreed
Signed	The current-state process map and the baseline sheet
Signer	The process owner
Exit gate	A map the people who do the work recognise, and a measured baseline
Typical duration	Three weeks

Mapping is where the lifecycle meets the real work for the first time. The aim is a picture of how the loop runs today, as the people who do it would describe it, together with numbers for what it costs. Both will be used for months. The map tells the next steps what to strip and sort. The baseline is what every later claim of improvement is measured against.



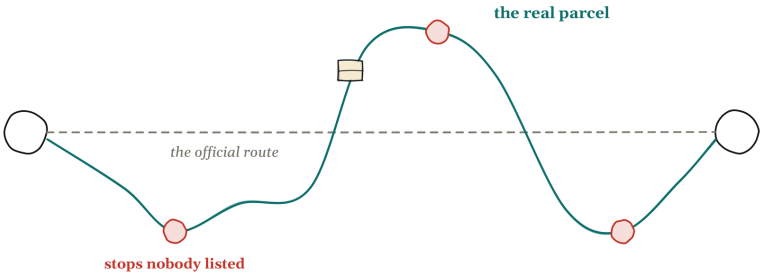
Like the before photo of a renovation: *taken of the room as it really is, so that every later claim of improvement is held up against it.*

The work

The map is drawn as swimlanes. Each lane is a role, a team, or an outside party such as a supplier or a customer. The steps run from left to right in the order they happen, each in the lane of whoever does it. Every step records the system or channel it uses, how many minutes of hands-on work it takes, and how long the case waits before the next step starts.

Two habits make the map honest. Draw it with the people who do the work, not the people who manage it. Managers describe the process as designed, and operators describe it as it runs. And walk real cases through it. Pick five recent cases, including one that went wrong, and follow each from start to finish. The steps the official map leaves out appear in the first hour.

While mapping, mark each step that looks like a workaround: re-keying between systems, printing for a signature, a batch job that runs weekly, an email chase between teams. Mark each step that redoes something that should have been right the first time. Do not argue about them yet. That is Step 3's job.



Like following five real parcels with a tracker: *the official route is a straight line, and the extra stops nobody listed show up within the first hour.*

The baseline

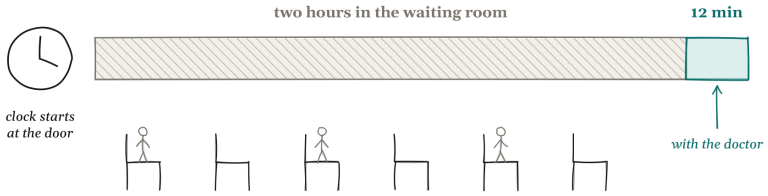
The baseline is the part most programmes skip and later regret. It records, for a sample of real cases, five numbers. These are the volume, the cost per case, the elapsed time from start to finish, the share of cases that needed an exception, and the share that were right the first time.

Cost per case should be fully loaded. It counts the people's time at their real cost, the systems, the rework, and the cost of errors that reach the outside. Elapsed time should be measured from the moment the case arrives, not from when someone first touches it. The waits are usually most of the time.

Two derived numbers help. Hands-on time divided by elapsed time is the flow efficiency of the loop. In most office processes it is shockingly low, often under 5%. The count of handoffs, each time the case moves between lanes, tells you where delay and errors gather.

The baseline must be measured, not estimated. Estimates are almost always too optimistic about the current process, which makes later savings look

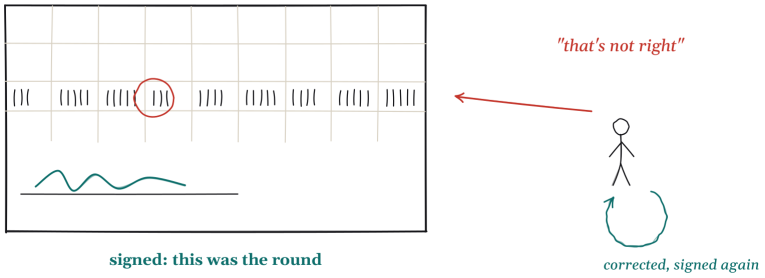
smaller than they are. Or they are too pessimistic, which makes the business case look better than it will turn out to be. Either way the economics at Step 8 will rest on sand.



Like a clinic visit timed from the moment you walk in: *twelve minutes with the doctor, two hours waiting, so the waits are most of the time.*

What gets signed

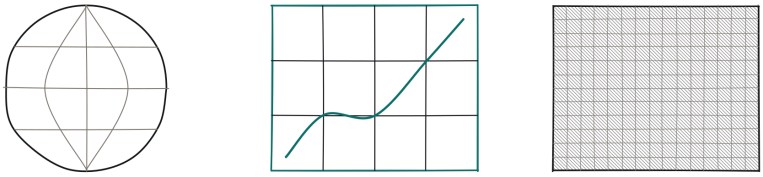
The process owner signs the map and the baseline sheet. The signature says two things: the map is how the work runs, and the baseline is the number we will be held to. If an operator later says "that's not how we do it", the map goes back to be fixed, and the owner signs again.



Like a golfer signing the scorecard: *it says these were the strokes, and if a marker spots a wrong one, it is corrected and signed again.*

How it fails

The most common failure is mapping the official process. The procedure manual is copied into lanes, nobody walks a real case, and the workarounds stay invisible. A second is mapping at the wrong grain. A map with three boxes hides everything, and a map with ninety boxes cannot be read. Somewhere between eight and twenty steps is usually right for one loop. A third is skipping the baseline because "we know roughly what it costs". Nobody does.



Like choosing a map for a city walk: *a globe hides every street, a plan of every brick cannot be read, and the street map in between works.*

What the foundations add

A map means something only when its structure mirrors the work. Douglas Hofstadter called this isomorphism: symbols carry meaning when they change as the thing they describe changes. A map copied from the procedure manual mirrors the manual, not the work. That is why the map is drawn from walk-throughs of real cases and checked against the system log.

Mapping is also a choice of level. The same invoice can be described as keystrokes, as steps, as decisions or as one line in the accounts. Keystrokes are too fine to redesign, and the accounts are too coarse. The useful level for embedding is steps and the decisions inside them, because that is the level at which work can be stripped, sorted and handed to an executor. Every map leaves things out. The test is whether what it leaves out would have changed the design.



Like a mirror and a portrait: *the mirror moves as you move, the portrait keeps an old pose, so the map is drawn from the work itself.*

At the manufacturer

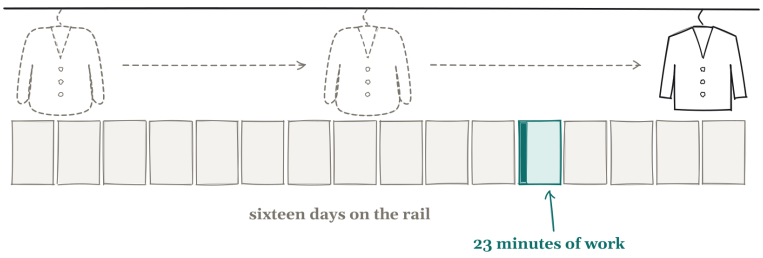
The invoice loop had six lanes: the supplier, a scanning desk, the accounts payable clerks, the budget owners across the plants, the financial controller,

and treasury. Two half-day sessions with four clerks and a team lead drew fourteen steps.

The walk-through of real cases changed the map twice. The official procedure said invoices arrived through the supplier portal. In fact, about half still arrived as PDF attachments, and a clerk printed and scanned them so the scanning desk could read them into the system. The procedure said budget owners approved in the ERP. In practice, most printed a batch of invoices every Friday, signed them, and sent them back by internal mail.

Finance measured 1,200 invoices over six weeks. The fully loaded cost was €11.80 per invoice. Elapsed time from receipt to payment-ready averaged sixteen days. Twenty-four percent of invoices raised an exception, and 71% matched their order and receipt the first time. Hands-on work averaged twenty-three minutes per invoice, which put flow efficiency well under 1%. Nine clerks did the work.

The numbers sat within published ranges. Industry surveys put the average cost of processing an invoice at around nine to ten dollars, with the best performers far lower. The baseline was not unusual. It was simply measured for the first time. The head of accounts payable signed the map and the baseline sheet.



Like a coat left at the dry cleaner's for sixteen days that gets twenty-three minutes of cleaning: *the invoice spent almost all its time waiting.*

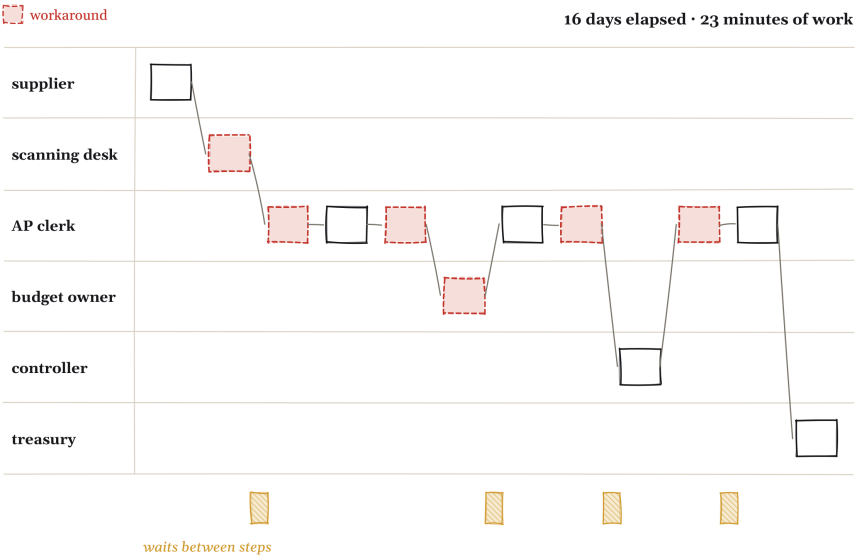


FIGURE 10 Step 2, Map. The loop drawn as it really runs: six lanes, fourteen steps, workarounds marked in red, and long waits between steps. Twenty-three minutes of work take sixteen days.

PART THREE

Redesign

XI

Step 3: Strip

The step at a glance	
Question	Which steps exist only because of constraints that no longer apply?
Starts when	The current-state map and the baseline are signed
Signed	The strip register
Signer	The process owner
Exit gate	Every step on the map is marked struck or kept, with the constraint named
Typical duration	One week

Every documented process is a mixture of three things. The first is the outcome itself. The second is the steps the outcome genuinely needs. The third is a layer of workarounds for constraints that once applied and no longer do. A batch step exists because a system once ran overnight. A handoff exists because two teams once sat in different buildings. A second review exists because errors were once frequent. A printed form exists because approvals were once on paper.

These workarounds are called compensations, because each compensates for a constraint. Over years they become indistinguishable from the real work. People are trained in them, controls are built on them, and optimisation polishes them until they look essential. Attach an executor that decides to a process full of compensations, and it will run the workarounds faster. It will also keep every wait between them.

Strip comes before any rule is written down, for that reason. There is no point gathering the rules of a step that should not exist.



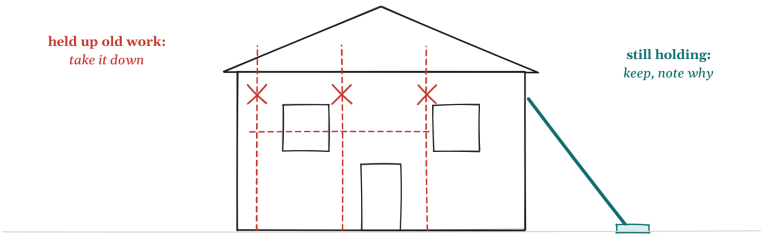
Like a path that still bends round trees felled years ago: *walkers keep every detour, and a faster walker simply takes the same bends sooner.*

The work

Take the signed map and walk through it step by step. For each step, ask one question: what constraint is this step compensating for, and does that constraint still apply?

The answer falls into three groups. Some steps serve the outcome directly: matching an invoice to its order is part of approving it. Keep these. Some compensate for a constraint that is gone: re-keying a PDF exists because the systems were not connected, and they now can be. Strike these. Some compensate for a constraint that still applies: a four-eyes check on large payments compensates for the risk of fraud, which has not gone away. Keep these, but record the constraint, because Step 5 may meet it differently.

Record each step in the strip register with its status and the constraint behind it. The constraint is the important column. A step struck without a named constraint is an opinion. A step struck because "the weekly batch dates from the 2009 ERP, which now posts in real time" is a finding.

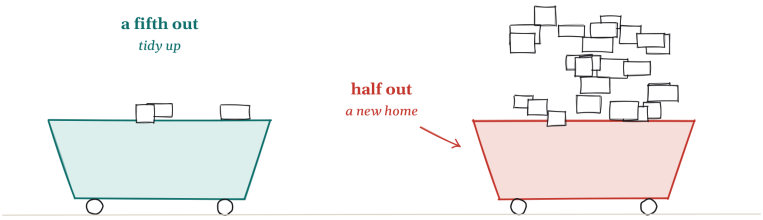


Like clearing scaffolding from a finished house: *take down the poles that held up work long done, keep the prop still holding a wall, and note why.*

The strip ratio

The share of steps struck is the strip ratio. It tells you what kind of work you are about to do.

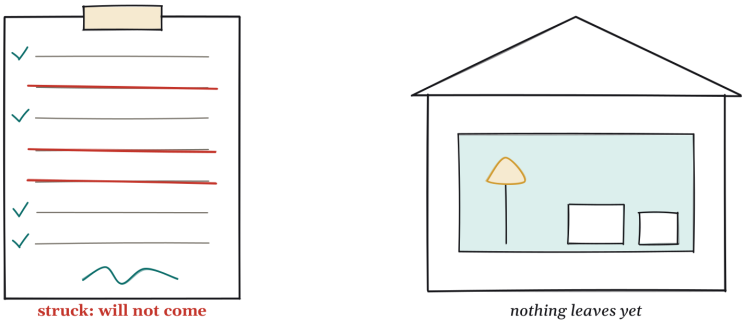
A low ratio, under a fifth, means the process is mostly real topology. Embedding it is largely a matter of letting an executor run the steps that remain. A high ratio, over a third, means the process is mostly accretion. Embedding it is a redesign, and the loop that comes out will look little like the map that went in. Neither is better. But a team that expects a light automation project and finds a 45% strip ratio needs to know before it plans the next three months.



Like clearing a house before moving: *if a fifth goes in the skip you tidy up, if half does you are planning a new home, so measure first.*

What gets signed

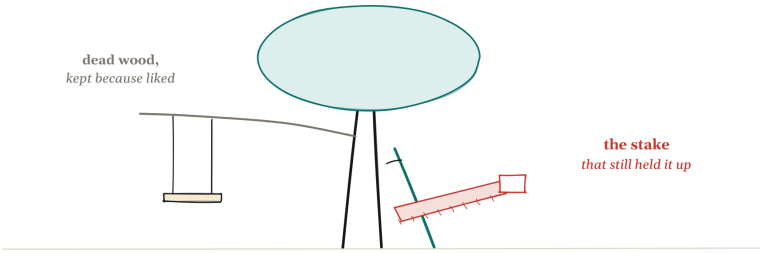
The process owner signs the strip register. Struck steps are not removed yet. Nothing changes in the live process at this step. The signature says that these steps will not be carried into the design, and that the owner accepts the reason given for each.



Like a packing list before a move: *the owner strikes what will not come to the new house, but nothing leaves the old one yet.*

How it fails

The usual failure is stripping too little because every step has a defender. Someone built the weekly reconciliation, and someone else relies on its output. The way through is to ask what the output is used for and whether the new loop could produce it directly. A second failure is stripping controls. A review that looks redundant may be the only thing standing between the company and a fraud. If the constraint still applies, the step stays, or its purpose moves to a junction or the standard. The third failure is quietly skipping the step, which leaves every compensation in place for the executor to inherit.

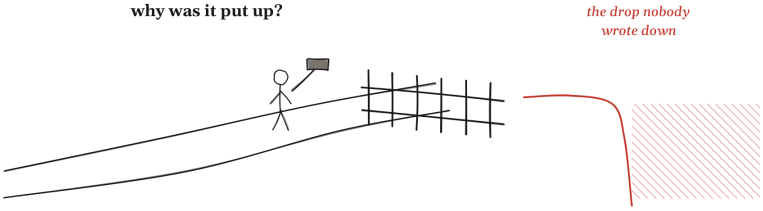


Like pruning a tree: *dead branches stay because someone hung a swing on them, while the stake still holding the tree up is sawn through.*

What the foundations add

Stripping is compression. The shortest description of a process that still predicts its results is the best understanding of it, and every compensation is a line in the description that predicts nothing anymore. It answers to a constraint that has gone. Removing it makes the rules that remain fewer and truer, which matters twice over once a model reads them. A model follows a long, inconsistent procedure as faithfully as a short, clear one, so it will faithfully automate the noise.

There is a discipline to it. The writer G. K. Chesterton advised against removing a fence until you know why it was put up. Each compensation is struck only after someone has named the constraint it answered and confirmed that the constraint is gone. A step struck without that check sometimes turns out to have been holding up a rule nobody wrote down, and Shadow will find it the hard way.



Like Chesterton's fence across a lane: *find out why it was put up, and confirm the reason has gone, before you take it down.*

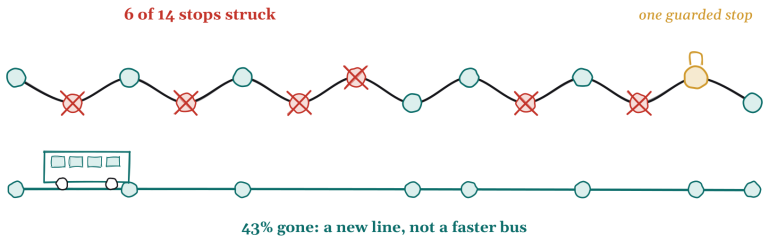
At the manufacturer

The clerks and their team lead went through the fourteen steps in a single afternoon. Six were struck.

Printing and scanning PDF invoices compensated for a scanning desk that predated email intake. The invoice platform could now read attachments directly. Budget owners' printed Friday approvals compensated for a time when approvals were on paper, and the ERP had supported electronic approval for years. Weekly batch posting compensated for an ERP job that had once run weekly and now ran continuously. A second clerk re-checked every invoice under €1,000. That step had been added after an error spike years before, and the errors had long since stopped. Clerks chased goods receipts by email because the receiving teams had no way to see which invoices were waiting. And a manual duplicate check against a spreadsheet compensated for a duplicate rule the ERP now applied itself.

Eight steps were kept. One of them, treasury's call-back to verify changed supplier bank details, compensated for fraud risk that was very much alive. It stayed, marked as a likely junction.

The strip ratio was 43%. That told the team it was redesigning the loop, not just speeding it up. The head of accounts payable signed the register.



Like an old bus route of fourteen stops: *six served places that had moved away, one guarded stop stayed, and what remained was a new line.*

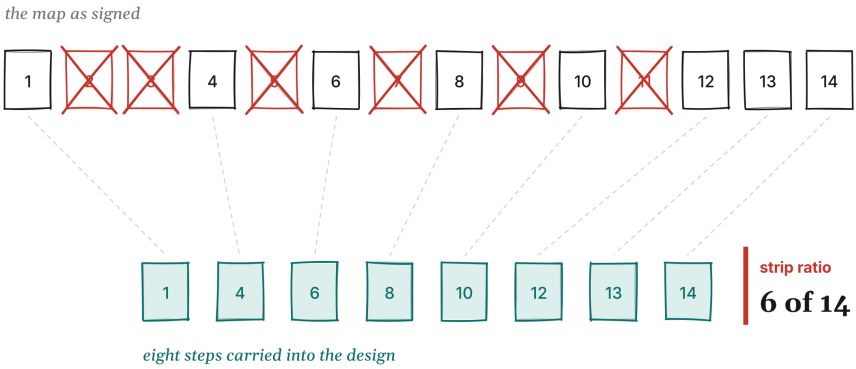


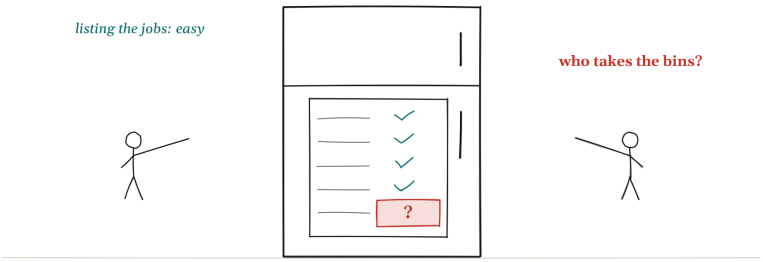
FIGURE 11 Step 3, Strip. Fourteen steps go in; six are struck, each with the constraint it compensated for. Eight remain. A strip ratio of 43% means the loop will be redesigned, not just sped up.

XII

Step 4: Sort

The step at a glance	
Question	Which rules apply, and who decides what?
Starts when	The strip register is signed
Signed	The rule register, the decision register and the contested-decision log
Signer	The process owner; the executive sponsor or the relevant leaders for contested decisions
Exit gate	Every rule owned and dated, every decision classed, no contested decision left open
Typical duration	Three weeks, of which the contested decisions often take most

Sort is where the lifecycle meets the company's unspoken agreements, and where the most important findings of the whole programme usually appear. It has two halves. The first gathers the rules the loop follows. The second sorts every decision the loop contains.



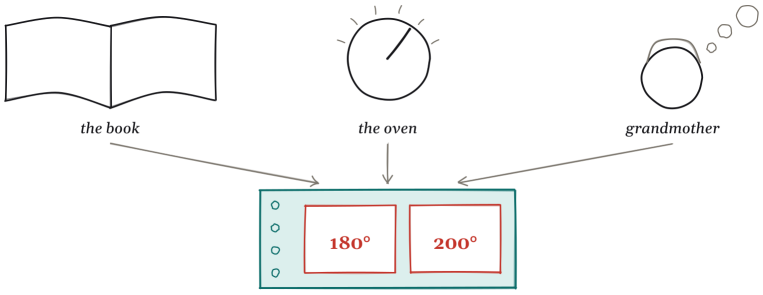
Like a house-share writing its chores down for the first time: *listing the jobs is easy, and the real argument is who takes out the bins.*

Gathering the rules

The rules of a loop live in three places. Some are documented in policies and procedures. Some are configured in systems, as tolerances, approval limits and validation checks. And some live in the heads of experienced staff: the supplier who always invoices in two parts, the cost centre that is never charged for freight, the threshold below which nobody really checks.

All three must be gathered into one rule register. Each rule gets a source, an owner, and a date. Rules in people's heads are written down in the words of the person who holds them, then confirmed by the owner.

Then compare. Where the policy and the system disagree, or two experienced clerks apply different rules to the same case, record a contradiction. **Contradictions are findings, not noise.** Each one is a place where the executor would have had to guess, and where the current process has been guessing for years. Each needs an owner to decide which rule stands.



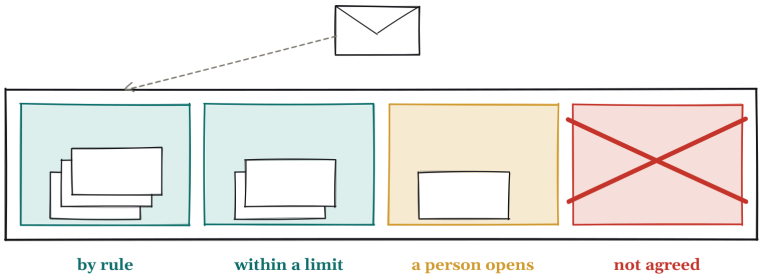
Like gathering a family's recipes from the cookbook, the oven settings and grandmother's memory into one binder, and finding two temperatures for the same cake.:

Sorting the decisions

Next, list every decision the remaining steps contain. A decision is any point where the work could go more than one way. Is this a duplicate? Does the price match within tolerance? Which cost centre? Approve or hold? Phrase each as a question.

Then sort each into one of the four classes. Mechanical decisions follow a rule completely. Bounded decisions need judgement inside a written limit. Consequential decisions carry a high cost of error or cannot be undone, so a person always decides. Contested decisions are ones the organisation has never agreed on.

The sort is partly technical and mostly organisational. A decision is mechanical only if its rule is written and uncontradicted. It is bounded only if someone has signed the limit. It is consequential because of its stakes, not because someone is nervous. And it is contested if, when the question is asked in a room, two senior people answer differently and both are sure they are right.



Like a mailroom's four pigeonholes: *letters handled by rule, ones settled within a limit, ones a person must open, and a fourth boarded up until agreed.*

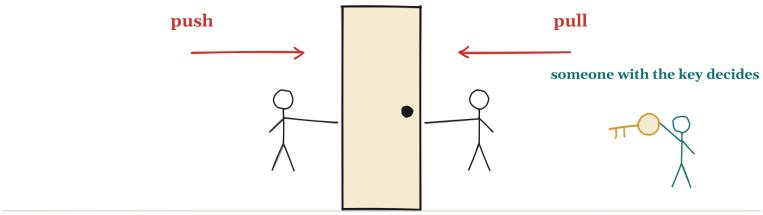
The contested decisions

Contested decisions deserve their own section because they decide the pace of everything after this step.

They are found by asking. Put each decision to the process owner and the people around them, and listen for disagreement. They are usually decisions that the old process handled by escalation, by habit or by not looking. Pay a disputed invoice to keep a key supplier happy? Give a goodwill refund outside the policy? Accept a late claim from a long-standing customer? Each has been answered case by case for years, differently each time.

No engineering can resolve a contested decision. An executor cannot be given a rule the company does not have. Someone with the authority to decide must decide, write the answer down, and sign it. In practice this is often a short meeting that has been avoided for years. It is also one of the most valuable meetings the programme will hold, because the same decision was going to surface as an incident the first time an executor met it.

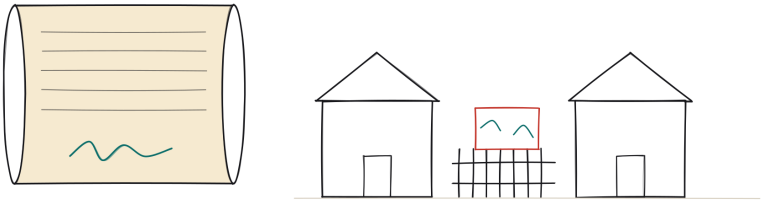
The exit gate of Sort is strict on this point. **A loop with an open contested decision cannot pass.** It can wait, and the rest of the work can continue. But nothing goes to Design until the contested-decision log shows every entry decided, by whom and when.



Like a door with one person pushing and another pulling: *no new hinge will fix it; someone with the key decides which way it opens.*

What gets signed

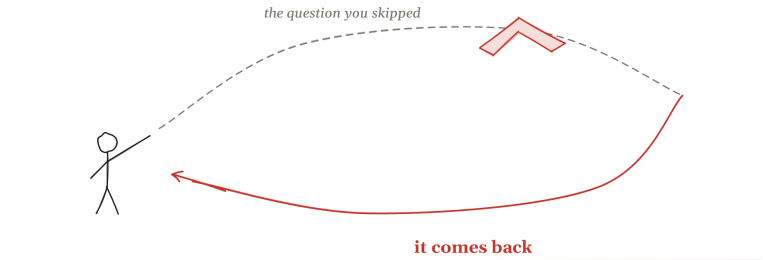
The process owner signs the rule register and the decision register. The contested-decision log is signed by whoever settled each entry, usually the executive sponsor or the leaders who share the decision.



Like selling a house: *the owner signs the deeds, while each boundary dispute is signed by the neighbours who settled it.*

How it fails

The first failure is treating Sort as documentation. A team collects the policies into a spreadsheet, does not interview the operators or read the system settings, and finds no contradictions because it never compared anything. The second is classing decisions by comfort. Everything is "consequential" because nobody wants to let go, so the executor is left with nothing to do. Or everything is "mechanical" because the pilot looked good, and the loop takes decisions nobody authorised. The third, and the most damaging, is noticing a contested decision and moving on. It will come back.



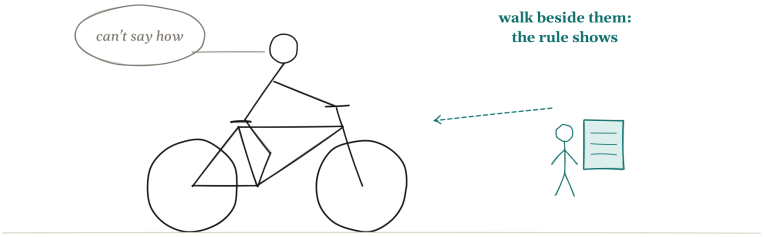
Like a boomerang: *throw an awkward question away unanswered and it comes back, usually the first time an executor meets it.*

What the foundations add

Sort is where the limits of rules become practical. Some rules can be gathered from documents and systems. Others live as tacit knowledge, and as the philosopher Michael Polanyi put it, people know more than they can tell. Asking experts to list their rules rarely works. Walking real cases with them does, because the rule shows up as a decision they make and can then explain.

The contested decisions are the deepest case. They cannot be settled inside the rule system, because the organisation never agreed on the answer. Gödel's result has a business analogue here: the system has to be extended from outside. Only a person with authority can do that, which is why Sort escalates rather than guesses.

The decision register matters for one more reason. Decisions written apart from the process steps can be given to an executor directly, tested one by one, and changed without redrawing the map. That separation is what makes the written half of the loop maintainable.



Like asking a cyclist how they keep their balance: *they cannot tell you, but walk beside them and the rule shows in what they do.*

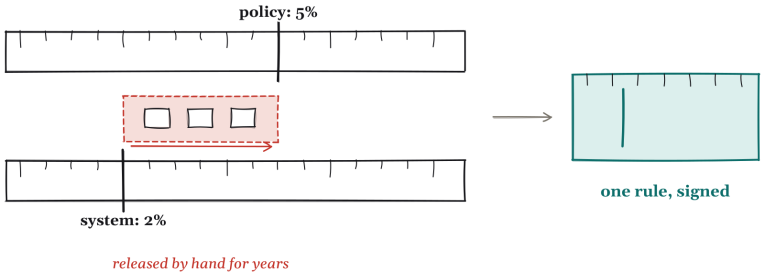
At the manufacturer

The rules came from four sources. These were the finance policy manual, the ERP configuration, the invoice platform's settings and three days of interviews with clerks. Fifty-eight rules made it into the register. Nine contradicted one another.

The largest contradiction was the price tolerance. The policy allowed a 5% difference between invoice and order before an invoice needed approval, while the ERP blocked anything over 2%. Clerks had been releasing blocked invoices between 2% and 5% by hand for years, which explained a good share of the exceptions. The controller decided the tolerance would be 2% for orders under €50,000 and 1% above, and the policy was changed to match. Smaller contradictions covered who could approve a non-order invoice under €500, and how to round invoices in foreign currencies.

Seventeen decisions remained after Strip. Eight were mechanical, such as duplicate detection, matching to an order and receipt within tolerance, and tax-code validation. Five were bounded, including coding a small non-order invoice to the right cost centre and accepting a partial delivery within limits. Three were consequential: releasing invoices over €50,000, accepting changed supplier bank details, and approving a non-order invoice over €5,000.

One was contested. When a supplier of a single-source part disputed a deduction and threatened to hold shipments, should the company pay the disputed amount to protect supply? Procurement said yes. Finance said never. Each had acted on its view for years. The CFO and the chief operating officer settled it in a thirty-minute meeting three weeks after it was logged. Payment was allowed up to €25,000, with the procurement director's signature and a record in the dispute log. Above that, it went to the CFO. The answer went into the contested-decision log with both their names.



Like two rulers marked at different lengths: *the policy said 5%, the system 2%, clerks bridged the gap by hand until the controller set one rule.*

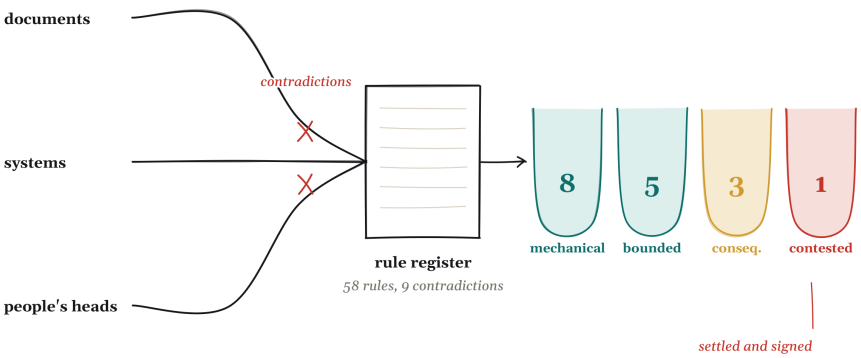


FIGURE 12 Step 4, Sort. Rules from documents, systems and people's heads flow into one register, where contradictions surface. Every decision is sorted into its class; the contested one is settled and signed before Design.

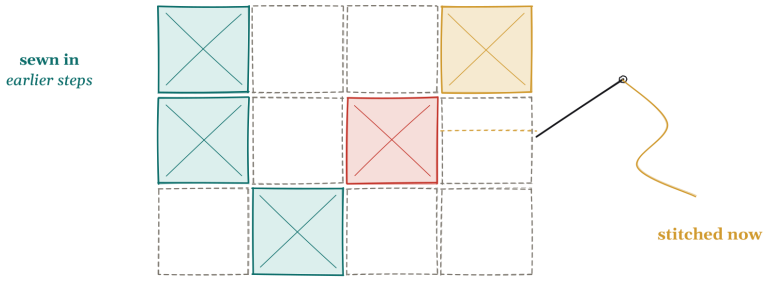
XIII

Step 5: Design

The step at a glance	
Question	What does the loop look like, including where people decide and how it is graded?
Starts when	The rule and decision registers are signed and the contested-decision log is closed
Signed	The Work Loop Canvas, the junction register and the standard
Signer	The loop owner
Exit gate	Gate G0 reached, with the nodes G1 needs signed
Typical duration	Two weeks

Design turns the findings of the first four steps into a loop. Up to now the lifecycle has described the old work and decided what to keep. Now it draws the new work: what the executor reads and changes, where people decide, where unhandled cases go, how the loop connects to its neighbours, and how its results will be graded.

The design is captured on the Work Loop Canvas. Several of its twelve nodes are already filled from earlier steps. The outcome came from Read. The compensations came from Strip. The rules and decisions came from Sort. Design fills the rest and signs the ones the next gate needs.



Like a quilt of twelve patches: *some were already sewn in earlier steps, and design stitches in the rest so the new work is whole.*

The work

Context comes first. List everything the executor must read to act well: the order, the receipt, the contract price, the supplier record. For each, name the system that holds it and the person who keeps it accurate. Mark any source the executor cannot reach, or that exists only as a scan. A gap here becomes a wrong decision later.

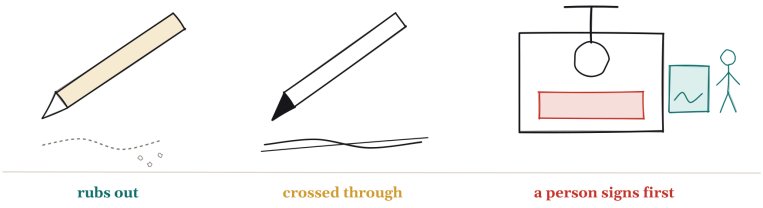
Actions come next. List everything the executor may change, in which system, with which permission. For each, answer one question: can it be undone, and at what cost? Reading is cheap. Posting an invoice can be reversed. Releasing a payment cannot.

Then place the junctions. Every consequential decision and every action that cannot be undone needs one. For each, write down who decides, what evidence the executor puts in front of them, and how a wrong decision is reversed. Then name the person who can switch the whole loop off, and how they would do it.

Then design the exception queue. Some cases will fall outside every rule. They go to a single queue with a named owner and a time limit. A weekly review looks at what landed there and turns repeated exceptions into new rules.

Then the handoffs. Write down what the loop receives, from which loop, and what it passes on, to which loop. For each, name who resolves a mismatch when the two sides disagree about what was sent.

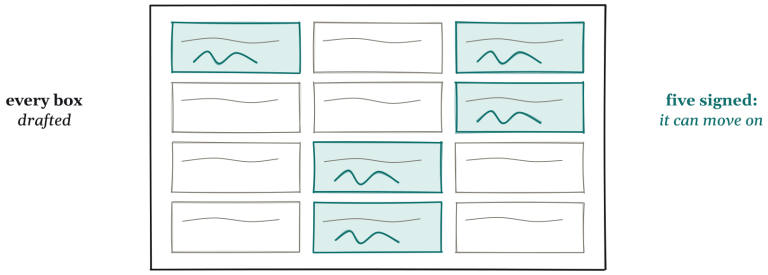
Finally, the standard. Write down how every outcome will be graded. A standard has criteria, the things an accepted outcome must satisfy, and a grading method. The usual pattern combines rules that check every case, a model that grades every case against the criteria, and a small human audit that keeps the model grader honest. The audit should look harder at the tail.



Like pencil, pen and printing press: *pencil rubs out, ink can be crossed through, but before the press runs a person signs the proof.*

Gate G0

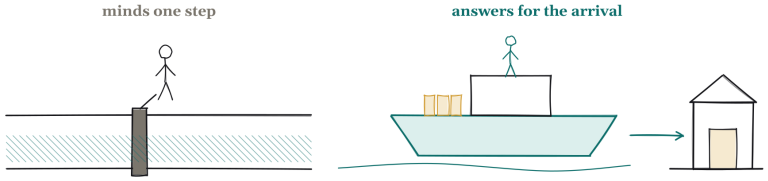
When every node on the canvas has a draft, the loop has reached G0, Mapped. For the loop to pass the next gate, five nodes must also be signed: the outcome, the compensations, the rules, the context and the standard. Design is the step where those signatures happen.



Like an application form: *every box filled in makes it complete, but it moves on only when the five boxes that need a signature are signed.*

What gets signed

The loop owner signs the canvas, the junction register and the standard. In many companies the process owner becomes the loop owner at this point. It is a small change of title that matters. A process owner looks after steps. A loop owner answers for an outcome, including the parts an executor produces.

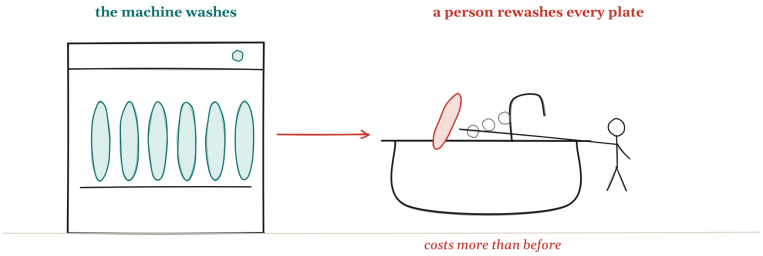


Like a lock-keeper becoming a ship's captain: *one minds a single step of the canal, the other answers for the cargo arriving at port.*

How it fails

The first failure is designing for the demo. The loop is drawn around what the pilot did well, and the junctions and exceptions are left for later. Later never comes. The second is placing junctions by comfort instead of by reversibility. Every step gets a human approval "to be safe", the executor does nothing a person does not repeat, and the loop costs more than the old one. The third is writing a standard that grades activity instead of outcomes. "Invoices

processed per hour" is not a standard. "Correct supplier, amount, tax and cost centre, with no duplicate, ready within terms" is.



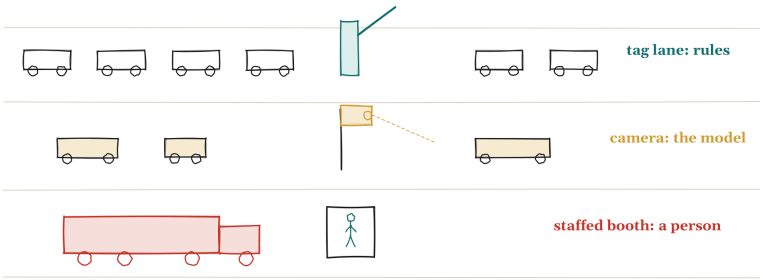
Like rewashing every plate the dishwasher cleaned, just to be safe: *the machine runs, nothing is saved, and the kitchen costs more than before.*

What the foundations add

Design allocates variety. Ashby's law says the loop as a whole must have as much variety as the cases it meets. Written rules take the body, where cases repeat and can be listed. The model takes the variety in between, reading cases that differ in ways no one could enumerate. People take the tail at the junctions. A design that asks rules to cover the tail becomes brittle. One that asks the model to cover it becomes unaccountable. One that sends everything to people has embedded nothing.

The strongest designs pair the two kinds of machinery inside each decision. The model proposes and the rules check: a match is suggested, then verified against the order and the receipt. A coding is drafted, then tested against the account rules. This is the pattern that made systems such as AlphaGeometry reliable, applied to ordinary work.

Design also builds the sensors with the actuators. The standard and the grader are designed at the same time as the actions, because a loop that cannot see its own results cannot be corrected. That is the closed loop drawn on paper, before anything runs.



Like a toll plaza: *tag lanes pass the regular cars, a camera reads the odd plates in between, and a staffed booth takes the rare oversized load.*

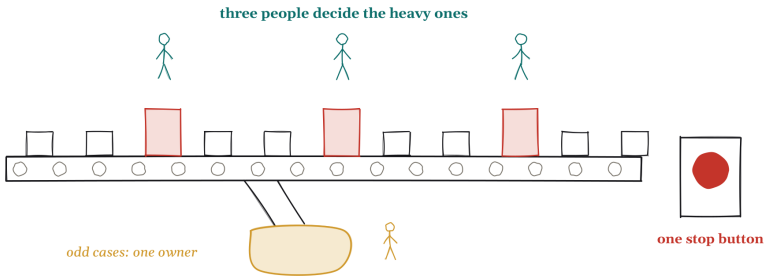
At the manufacturer

The executor would read the order, the goods receipt, the supplier record and contract prices. Contract prices turned out to live in PDF agreements, not in the ERP, for about a third of suppliers. Procurement agreed to load the prices for the top two hundred suppliers into the ERP before shadow running began. Below that, price checks would rely on order prices alone.

Three junctions were placed. Invoices over €50,000 went to the financial controller, who would see the matched order, the receipt, the contract price and the supplier's payment history. New or changed bank details went to a treasury analyst, who would call the supplier on a number held separately. Non-order invoices over €5,000 went to the budget owner of the cost centre. The head of accounts payable could switch the loop off from the platform's console. Invoices would then return to the clerks' queue within the hour.

The exception queue went to the accounts payable team lead, with a limit of two working days and a review every Monday. The standard required the right supplier, amount, tax and cost centre, no duplicate, and readiness within payment terms. Rules would check every invoice. A model grader would grade every invoice against the criteria. A finance quality analyst would audit 3% of invoices at random, and 20% of those over €10,000.

The head of accounts payable signed the canvas as loop owner. With the outcome, compensations, rules, context and standard signed, the loop stood at G0 and was ready for shadow running.



Like a packing line with three staffed checkpoints for the heaviest items, a reject bin with a named owner and one red stop button.:

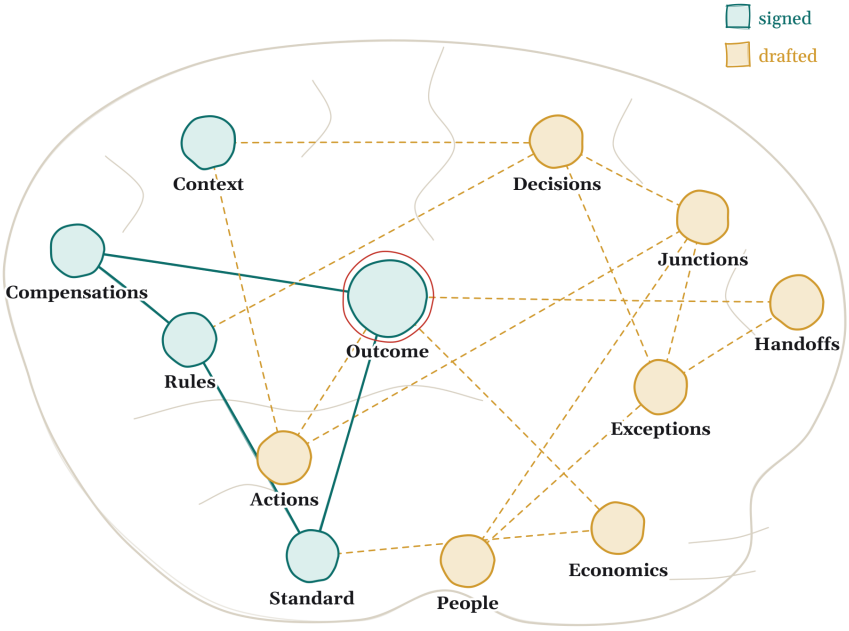


FIGURE 13 Step 5, Design. The canvas with every node drafted (G0, Mapped). The five nodes the next gate needs are signed: outcome, compensations, rules, context and standard.

PART FOUR

Prove

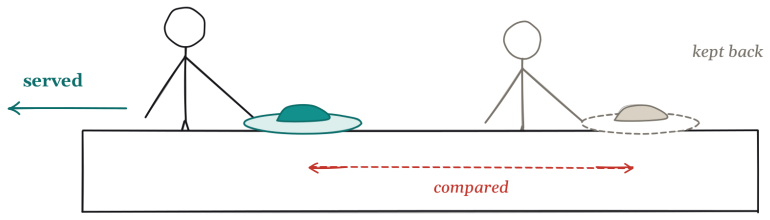
XIV

Step 6: Shadow

The step at a glance	
Question	Does the loop reach the same results as people, without acting on anything?
Starts when	Gate G0 is reached and the G1 nodes are signed
Signed	The shadow report
Signer	The loop owner and the grader
Exit gate	Gate G1 passed: agreement measured on the body and the tail, every disagreement explained
Typical duration	Four to eight weeks

Shadow is the first time the executor meets the real stream of cases. It reads every case the current process receives and reaches its own result. But it changes nothing. People keep doing the work as before. At the end of each case, the two results are compared.

The point is not to prove the executor is perfect. It is to learn, on real cases and at no risk, where it agrees with people, where it disagrees, and why.



Like a trainee chef cooking every dish beside the head chef: *only the head chef's plate goes out, and the two plates are compared each time.*

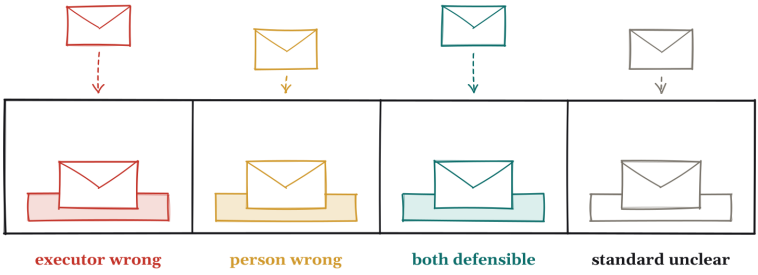
The work

Run the loop in parallel for long enough to see the whole distribution. A week of routine cases teaches little. The tail cases, which are rarer and more consequential, need several weeks to arrive in useful numbers. For most loops, four to eight weeks is enough.

Measure agreement separately for the body and the tail. A single overall number hides the most important fact. An executor can agree with people on 98% of routine cases and 80% of large ones, and the overall figure of 95% would describe neither.

Then study every disagreement. Each falls into one of four groups. The executor was wrong, and the cause is a missing rule or a gap in context. The person was wrong, which happens more often than teams expect, and it is worth recording. Both were defensible, which usually points to a rule that is vaguer than anyone thought. Or the standard itself was unclear, so the grader could not decide.

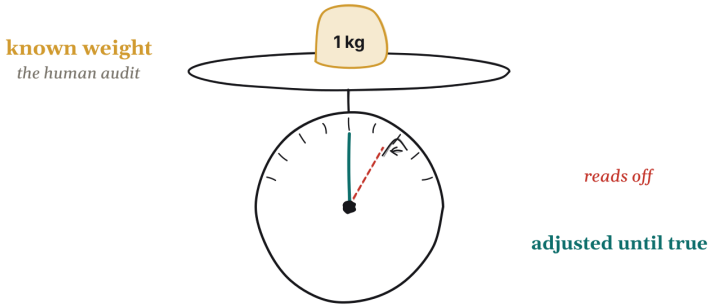
Each group sends the work somewhere. Missing rules go back to Sort, and the rule register is signed again. Context gaps go back to Design. Unclear criteria go to the standard. The shadow report records every change.



Like a post room with four pigeonholes: *every disagreement is filed by cause, and each hole is emptied by the desk that can fix it.*

The grader

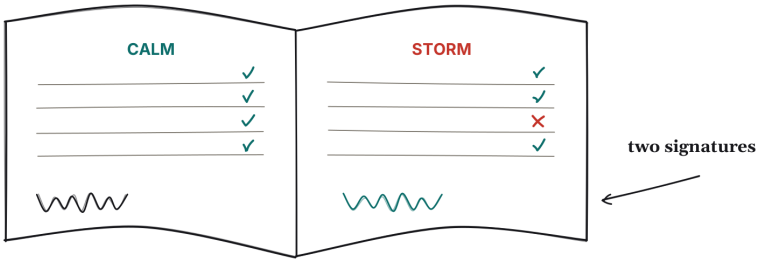
Shadow is also where the standard is tested. The model grader and the human audit both grade the executor's results and the people's results. When they disagree with each other, the grader is calibrated. By the end of shadow, the loop owner should trust the grader enough to rely on it, because from the next step onward the grader is the main way anyone knows the loop is working.



Like a market scale checked against a known weight each morning: *once its needle matches, traders trust it to weigh the whole day unwatched.*

What gets signed

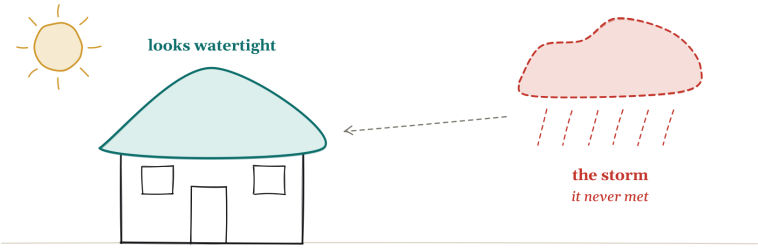
The loop owner and the grader sign the shadow report. The report states the agreement on the body and the tail, the disagreements by group, the changes made to rules, context and the standard, and the remaining known weaknesses. G1 is passed when the owner judges the evidence sufficient to let the loop act on part of the work.



Like a trainee pilot's logbook: *calm and stormy hours are tallied apart, the faults are listed, and two signatures release the next stage.*

How it fails

The first failure is too short a shadow. Two weeks of clean routine cases produce a reassuring number that says nothing about the tail. The second is reporting one overall agreement figure. The third is treating disagreements as the executor's fault by default. Teams that do this never discover the rules their own people apply inconsistently, which are often the more valuable finding. The fourth is changing the executor during shadow without recording why. When the loop later fails, nobody can say which version was proven.



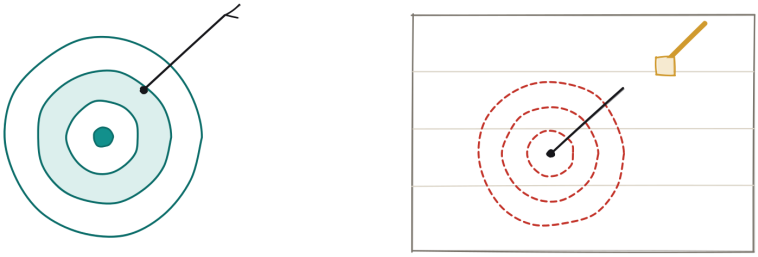
Like judging a new roof after two sunny weeks: *it looks watertight only because the storm that would test it never came.*

What the foundations add

Shadow is a test that can fail, and it should be set up that way. The agreement targets for the body and the tail are written into the G1 nodes before the first case arrives. Results that miss them stop the gate. Setting the bar after seeing the numbers turns the test into a description.

Shadow also maps what researchers call the jagged frontier. A learned model is strong and weak in places nobody can predict from a demonstration. Only the real distribution of cases shows where its edges are, band by band. That is why shadow runs on the live stream of cases, not on a curated set, and why the tail needs weeks to arrive in useful numbers.

Finally, shadow is where tacit rules surface. Every disagreement where the person was right and the executor wrong is a rule someone applied without writing it down. Every disagreement where the executor was right is a rule people had stopped following. Both feed the written half of the loop.



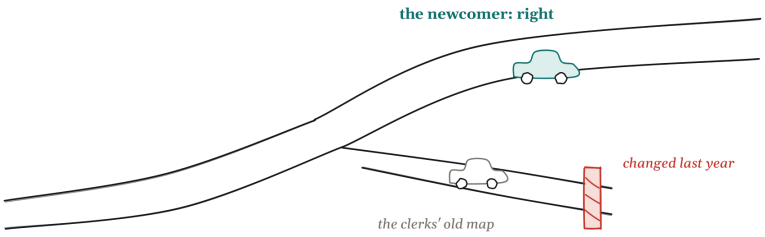
Like an archery target fixed before the shot: *painting the rings around wherever the arrow landed turns a test into a description.*

At the manufacturer

Shadow ran for six weeks on 9,800 invoices. Agreement with the clerks' results was 97.8% on invoices under €5,000 and 81% on invoices over €50,000. The overall figure of 94% appeared once in the report, with a note on why it should not be quoted alone.

Two kinds of error accounted for most disagreements. Credit notes from some suppliers looked like invoices, and the executor read several as invoices. And invoices covering more than one order confused the matching. Two rules were added to the register and signed again. Procurement corrected the supplier records that had caused one cluster of mismatches.

A third finding surprised the team. In about one in five disagreements on mid-sized invoices, the grader sided with the executor. Clerks had been applying an old cost-centre rule that finance had changed the year before. The rule was clarified and the clerks were briefed. The head of accounts payable and the finance quality analyst signed the shadow report, and the loop passed G1.



Like veterans driving by last year's road map: *in one disagreement in five the newcomer was right, because the clerks followed a rule finance had changed.*

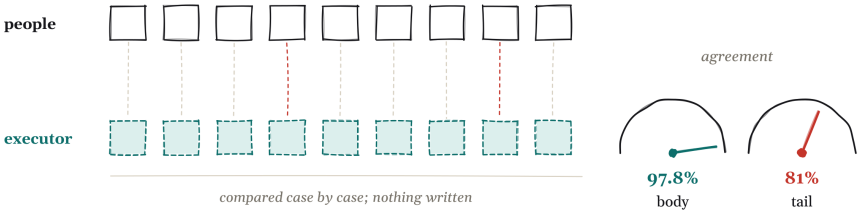


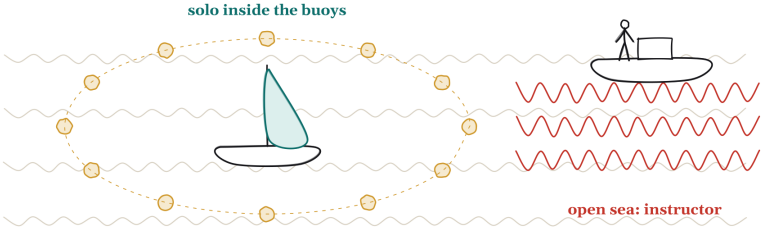
FIGURE 14 Step 6, Shadow. The executor and the people work the same cases side by side; nothing the executor does reaches the records. Agreement is read separately for the body and the tail.

XV

Step 7: Bound

The step at a glance	
Question	Can the loop act alone on part of the work, inside written limits?
Starts when	Gate G1 is passed
Signed	The gate review record
Signer	The control owner, with internal audit consulted
Exit gate	Gate G2 passed: the loop acts alone inside its limits, and the switch-off has been tested
Typical duration	Four weeks of bounded running before the review

Bound is the first time the executor acts. It does so only inside limits written down in advance, usually the routine body of the distribution. Everything outside the limits still goes to people. The step turns the evidence of shadow into a decision about trust, and it puts that decision in front of the person responsible for control.



Like a learner sailing solo inside a ring of buoys: *the routine water is theirs alone, while the open sea still needs the instructor aboard.*

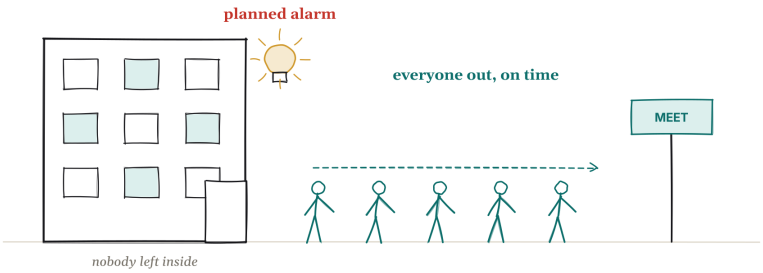
The work

Write the limits. They are drawn from the shadow evidence, not from hope. The cases where agreement was highest, the decisions that were mechanical or bounded, and the actions that can be undone define the first scope. Limits are usually expressed as plain conditions. Below a value. Within a tolerance. From a known counterparty. Of a known type.

Then complete the nodes the second gate needs. The actions, the decisions, the junctions and the exception queue must all be signed. The executor now writes to systems, so its permissions must match the actions node exactly. Permissions granted "for convenience" during the pilot are removed.

Then test the switch-off. Before the loop acts on its first live case, the person named as able to stop it does so, in a planned test. Cases should flow back to people within the stated time, and nothing should be lost. A switch-off that has never been tested is a hope.

Then run inside the limits for several weeks, grading every outcome, and bring the evidence to a gate review.

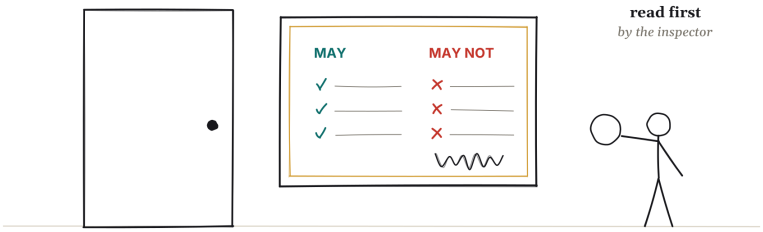


Like a fire drill before the building opens: *the alarm is pulled on purpose, everyone files out on time, and nobody is left inside.*

The gate review

The gate review is a short meeting with a written record. The control owner, such as the financial controller or the head of risk for the process, reviews the limits and the graded results. They also review the exceptions, the junction decisions and the switch-off test. Internal audit is consulted, so that the controls it will later test are the ones the loop actually has.

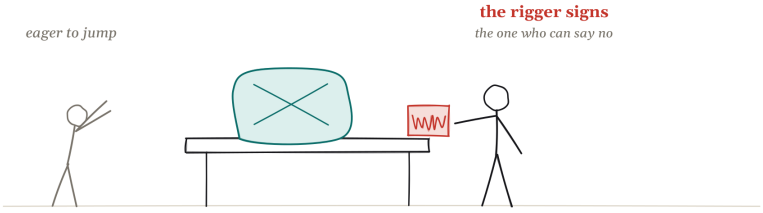
The record states what the loop may do alone, what it may not, and what evidence supported the decision. It is the document an auditor or regulator will ask for first.



Like a permit framed by a building's door: *it says plainly what may happen inside and what may not, and it is the first thing an inspector reads.*

What gets signed

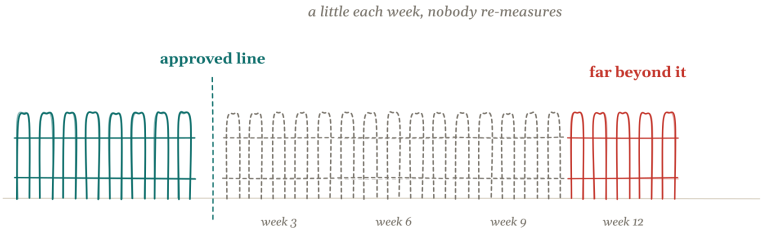
The control owner signs the gate review record. This is deliberately not the loop owner. The loop owner is responsible for the outcome and naturally wants the loop to run. The control owner is responsible for the company's controls. Bound is the step where those two views meet, and the signature belongs to the one whose job is to say no.



Like a parachute checked by the rigger, not the jumper: *the jumper is eager to go, so the signature belongs to the one whose job is to refuse.*

How it fails

The first failure is setting limits by ambition rather than evidence. The pilot handled large cases well in a demo, so large cases go into scope. The second is letting the scope creep. Each week someone adds another supplier, another value band, another type, and nobody holds a new review. Each extension of scope is a new gate, however small. The third is never testing the switch-off. The fourth is giving the executor broader system permissions than its actions node allows.

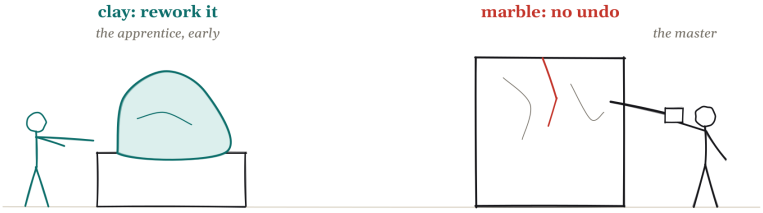


Like a garden fence moved out one section a week: *each move looks tiny, nobody measures again, and soon it stands far beyond the approved line.*

What the foundations add

The bounded scope is the written edge of the rule system. Inside it, the loop acts alone because the evidence says it can. Outside it, cases go to people. Drawing that line explicitly is the practical answer to the limits of rules: not pretending the rules cover everything, but saying exactly where they stop.

The line follows reversibility as much as value. An action that can be undone cheaply can be trusted to the loop earlier, because a mistake is a correction, not a loss. An action that cannot be undone, such as a payment, waits behind more evidence and a person at the junction. The control owner signs because their role is to stand outside the loop and say no. In Gödel's terms, they are the authority from outside the system that decides when it may be extended.



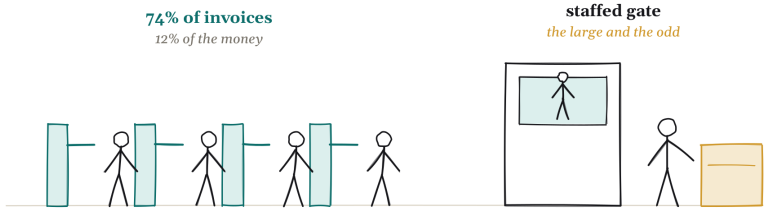
Like a sculptor's studio: *the apprentice may shape clay early because a mistake can be reworked, while the cut in marble waits for the master.*

At the manufacturer

The limits followed the shadow evidence closely. The loop could act alone on invoices backed by an order, under €5,000, within the 2% tolerance, from suppliers the company had paid for at least six months. That was 74% of all invoices by count, and about 12% of spend. Everything else stayed with the clerks and the junction holders.

The platform's permissions were cut back to the actions on the canvas. A service account from the pilot that could edit supplier records was removed. In the first week, the head of accounts payable switched the loop off during a planned test. Invoices returned to the clerks' queue in twenty minutes, and none was lost.

After four weeks the financial controller held the gate review. The graded results showed 98.6% of bounded invoices met the standard, and the exception queue had cleared within its limit every week. Internal audit asked for one change: the dispute log required by the contested decision was to be linked to the payment record. The controller signed the record, and the loop passed G2.



Like ticket gates at a station: *most travellers pass the automatic gates alone, while the few with heavy or odd luggage go to the staffed gate.*

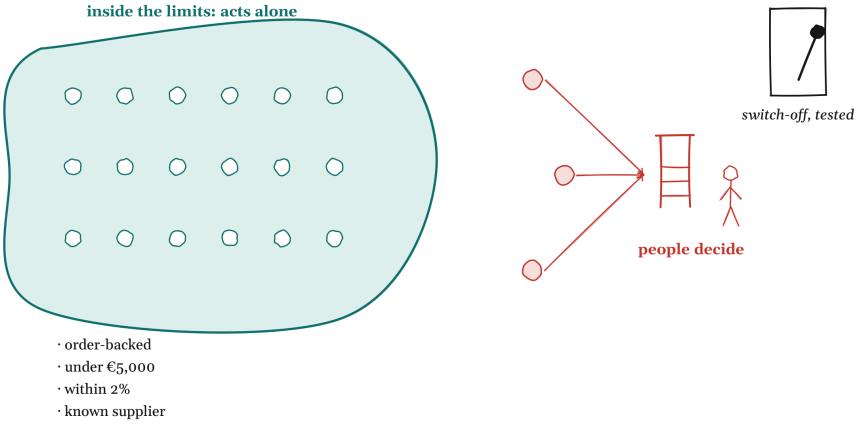


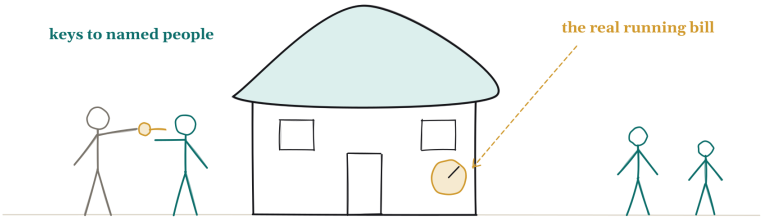
FIGURE 15 Step 7, Bound. The executor acts alone only inside a fenced scope drawn from the shadow evidence. Everything outside goes through gates to people, and the switch-off is tested before the first live case.

XVI

Step 8: Own

The step at a glance	
Question	Are named people in place, and what does one accepted outcome really cost?
Starts when	Gate G2 is passed
Signed	The loop charter and the economics sheet
Signer	The loop owner for the charter; finance for the economics
Exit gate	Gate G3 passed: people named and in post, cost per accepted outcome measured against the baseline
Typical duration	Six to ten weeks, mostly measurement

Own is the step that makes the loop part of the organisation rather than a project inside it. It does two things. It puts named people into every role the loop needs, and it measures, over enough weeks to trust the number, what the loop costs per accepted outcome compared with the baseline from Step 2.



Like a house handed from builder to household: *the keys pass into named hands, and the real running bill is measured month by month.*

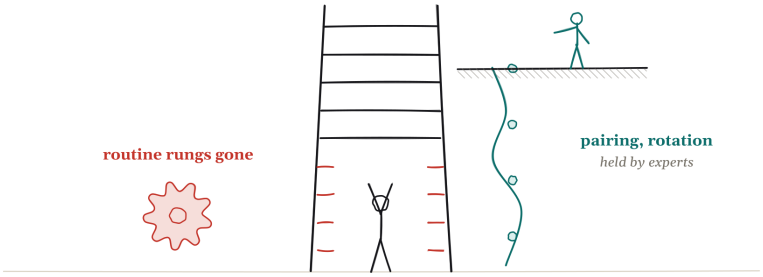
The people

A loop running at scale needs five kinds of role filled by named people, not by job titles.

The loop owner answers for the outcome. The junction holders decide the consequential cases. The exception owner works the queue and runs the weekly review. The grader keeps the standard calibrated. And someone owns adoption: the people whose work changes, how they move into new roles, and how new staff will learn the work when routine cases no longer teach it.

That last point is easy to miss. In the old process, junior staff learned by handling simple cases until they were ready for hard ones. When the executor takes the simple cases, that path disappears. The charter has to say how people will learn the judgement the junctions need. Common answers are pairing with junction holders, rotating through the exception queue, and reviewing graded cases.

The people part of Own also includes the plain question of whose job changes and how. This is better answered openly and early than discovered by rumour. Most loops move people from routine work to exceptions, junctions, supplier relationships or the next loop to be embedded. Some roles shrink. The charter records the changes agreed, and the role change plan at Step 10 carries them across the map.



Like a ladder whose lowest rungs the machine has taken: *new staff need another way up to the hard cases, such as a rope held by experts.*

The economics

The economics compare the loop's cost per accepted outcome with the baseline. Each part of that phrase matters.

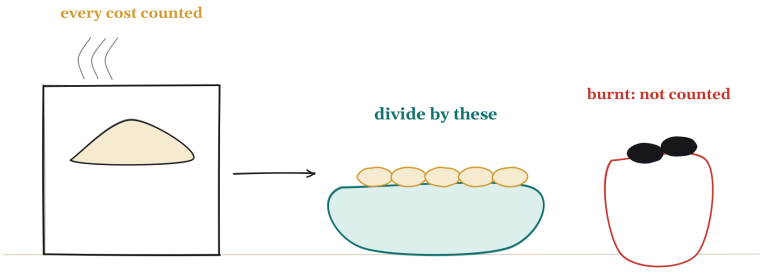
Cost is fully loaded, as the baseline was. It includes the model and platform costs, the people at junctions and in the exception queue, the grading and audit, and the rework. Leaving out the people still working in the loop is the most common way to flatter the number.

Per accepted outcome means dividing by the outcomes that met the standard, not by the cases processed. A loop that processes many cases quickly but sends a fifth back for rework is not cheap.

Compared with the baseline means against the measured figure from Step 2, over a comparable period. And the body and the tail are reported separately. A

loop can cut the cost of routine cases sharply and leave the tail unchanged, and the average would hide both facts.

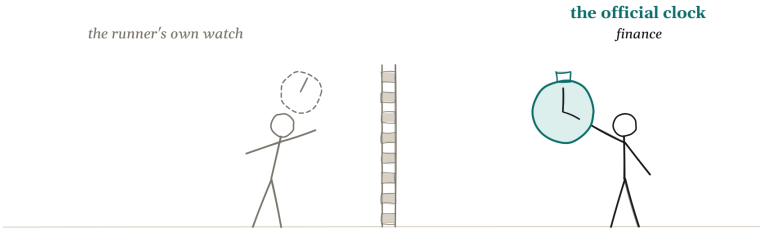
Measure for long enough to cover a full business cycle where possible, such as a month-end or a quarter-end. Six to ten weeks is typical.



Like pricing a bakery's bread by loaves that pass, not loaves baked: *count flour, oven and staff, then divide by the good ones only.*

What gets signed

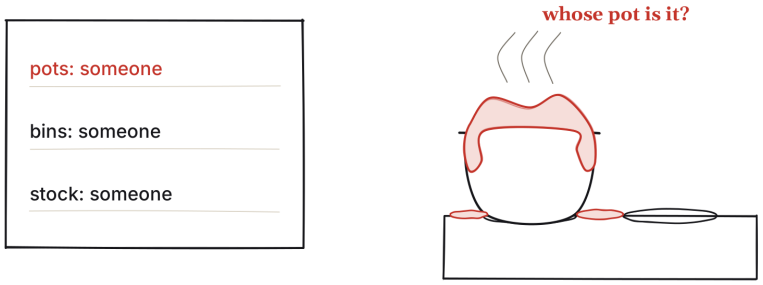
The loop owner signs the loop charter, which names every role and records the agreed changes to people's work. Finance signs the economics sheet, usually the planning and analysis lead. It is finance's number, not the programme's.



Like a race timed by the official's stopwatch, not the runner's wristwatch: *finance holds the clock, so the cost figure is theirs, not the programme's.*

How it fails

The first failure is naming roles, not people. The charter says "junction holder: finance" and nobody in finance knows they hold it. The second is flattering the economics. The model cost is counted, but the exception queue, the grading, the audit and the platform are not. The third is skipping the adoption owner, which leaves the change to people's work unmanaged and the next generation of experts untrained. The fourth is declaring savings that depend on headcount changes nobody has agreed.

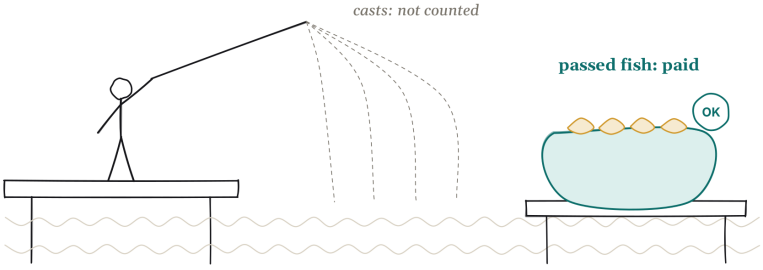


Like a duty rota that says 'someone' on every line: *when the pot boils over, nobody knows it is theirs to deal with.*

What the foundations add

Named people are part of the regulator, not decoration around it. When the loop meets something its rules cannot handle, a person extends the rules. The exception owner turns a recurring exception into a new rule. The junction holder decides a case no rule anticipated. The loop owner settles what the register leaves open. A loop with no named owners has no way to grow its written half, so it slowly falls behind the work.

Cost per accepted outcome is the feedback signal for economics. It counts only results that met the standard, so it cannot be improved by doing more work badly. The economist Charles Goodhart warned that a measure made into a target stops measuring. Activity counts, such as cases processed, fail that test. Accepted outcomes, counted against a measured baseline, hold up.



Like paying an angler by fish that pass the market inspector, not by casts: *casting more cannot raise the pay, only good catch can.*

At the manufacturer

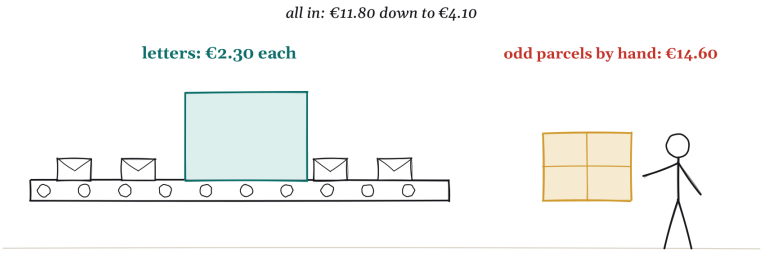
The charter named the head of accounts payable as loop owner. The financial controller, a treasury analyst and fourteen budget owners were named as

junction holders. The accounts payable team lead owned the exception queue, and the finance quality analyst the grading. The finance operations manager took on adoption.

The changes to people's work were agreed with the clerks over three weeks. Of the nine clerks, four moved to work the exception queue and supplier queries. Three moved to the supplier onboarding team, which the map had marked as a loop to embed later. One moved to treasury. One retirement during the period was not replaced. New clerks would rotate through the exception queue and sit with the controller for junction decisions during their first three months.

Finance measured eight weeks across a quarter-end. The fully loaded cost per accepted invoice was €4.10, against the baseline of €11.80. That figure included the platform, the model usage, grading, audit and every hour spent at junctions and in the exception queue. Reported separately, invoices in the body cost €2.30 each and those in the tail €14.60, because the tail still needed people. Elapsed time from receipt to payment-ready fell from sixteen days to 3.4. The share right first time rose from 71% to 93%.

The head of accounts payable signed the charter. The planning and analysis lead signed the economics sheet. With the people and economics nodes signed, the loop passed G3.



Like a sorting office after the machine arrived: *each letter costs far less all in, while odd parcels still go by hand and cost the most.*

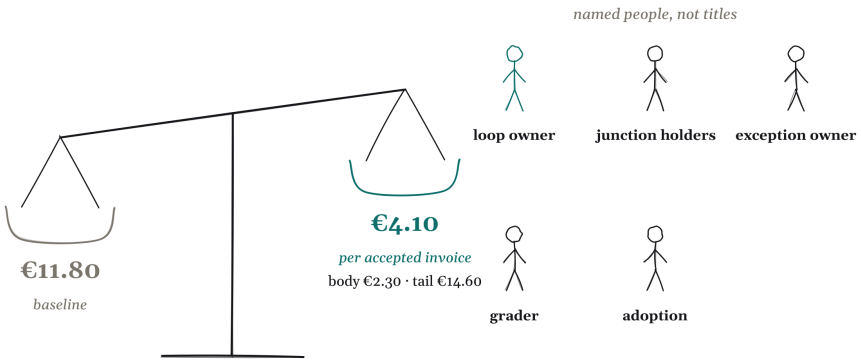


FIGURE 16 Step 8, Own. Named people fill every role the loop needs, and the cost of one accepted outcome is weighed against the measured baseline, body and tail apart.

PART FIVE

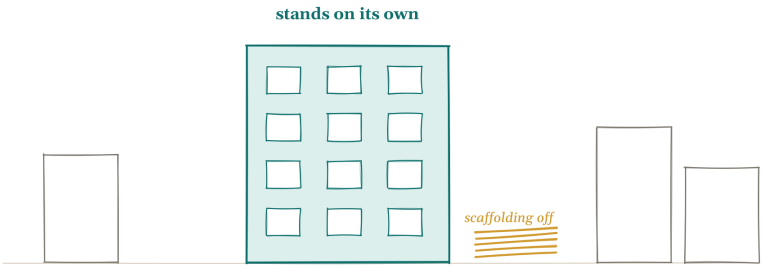
Run and Scale

XVII

Step 9: Core

The step at a glance	
Question	Does the loop run as part of the business, and would we know if it stopped working well?
Starts when	Gate G3 is passed
Signed	The operating review, each month, and the switch-off plan
Signer	The loop owner
Exit gate	Gate G4 passed: all twelve nodes signed, the first operating reviews held, the switch-off tested on schedule
Typical duration	Ongoing; G4 is usually confirmed after the first two or three monthly reviews

Core is the step where the loop stops being a programme and becomes how the work is done. Its full scope is switched on. The old process is retired. From here on, the loop is run, measured and reviewed like any other part of the business. The difference is that an executor that decides is doing much of the work, so the reviews have to watch for things an instruction-follower would never do.

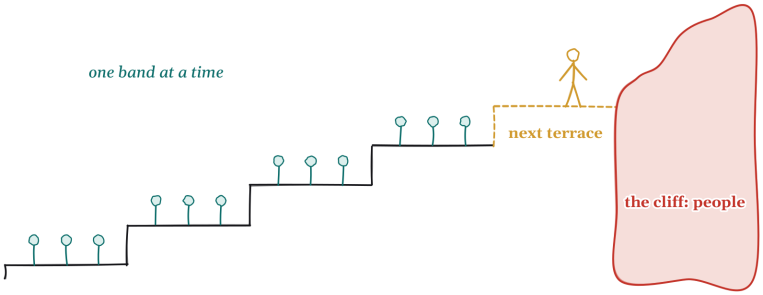


Like a finished building with its scaffolding packed away: *it now stands on its own and is looked after like every other building on the street.*

Widening the scope

Core usually begins by widening the limits set at Bound, one band at a time. Each widening is small, evidenced and recorded. It might mean larger values, newer suppliers, other case types. A widening is a gate in miniature, with the control owner's agreement. The tail stays with the junctions. Core does not

mean the executor decides everything. It means the loop, people included, runs the whole outcome.



Like a vineyard adding one surveyed terrace a season: *each new step is checked before planting, and the steep cliff above is left to people.*

Drift

Every loop that runs long enough will drift. Its inputs change: a supplier redesigns its invoices, a new product line appears, a regulation changes a tax rule. The model behind the executor may be updated by its provider. People at the junctions get used to agreeing with the evidence and stop reading it closely. None of these shows up as an error message. They show up as a slow change in the share of outcomes that meet the standard.

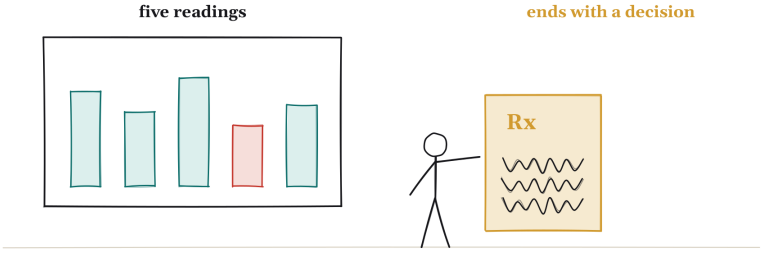
Drift is caught by grading every outcome, as the standard requires, and by a weekly sample in which a person compares a handful of the executor's results with their own judgement. The grading shows when quality moves. The sample shows when the grader itself has started to miss things.



Like a clock that loses a minute a week: *it never stops or beeps, so only setting it against the radio time signal shows the slow error.*

The operating review

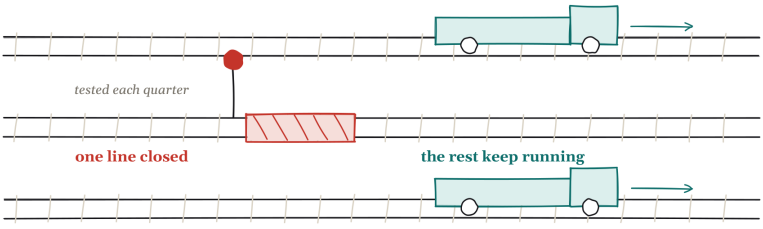
Once a month, the loop owner holds an operating review. It looks at five things. The first is the share of outcomes meeting the standard, body and tail separately. The second is the cost per accepted outcome. The third is the exception queue, including the oldest case and the repeated exceptions that should become rules. The fourth is the junction decisions, including how often junction holders overrode the executor's evidence. The fifth is any drift found. Each review ends with decisions, which are recorded. A review that ends without decisions is a status meeting.



Like a monthly check-up: *five readings are taken, and the visit is worth having only if it ends with something written on the prescription pad.*

The switch-off plan

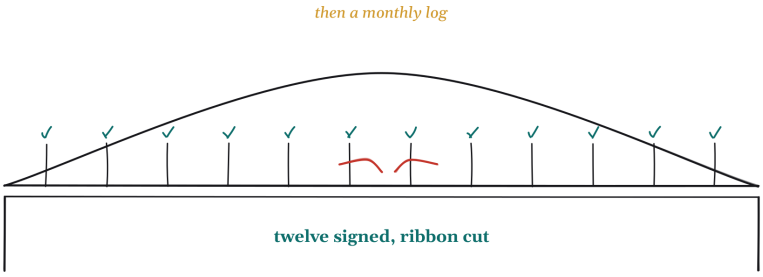
The switch-off was tested once at Bound. In Core it becomes a plan. It records who can stop the loop, how, how quickly cases return to people, and what happens to cases in flight. It is tested on a schedule, usually every quarter. It is also used, without hesitation, when drift or an incident makes the executor's results untrustworthy for a part of the work. Switching off a part of the scope for a few days is normal operations, not failure.



Like a railway closing one line for repairs while trains run on the others: *it is practised every quarter and done without hesitation when needed.*

What gets signed

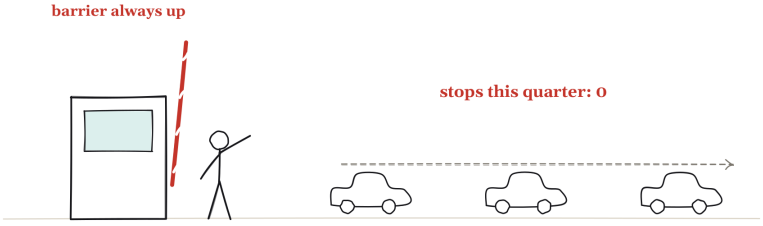
The loop owner signs each monthly operating review and the switch-off plan. With all twelve canvas nodes signed and the first reviews held, the loop passes G4. That is the formal end of the embedding. The loop is in the core.



Like a new bridge at its opening: *all twelve inspections are signed, the ribbon is cut, and from then on a monthly log keeps it in service.*

How it fails

The first failure is declaring victory at go-live. The programme team disbands, the reviews stop after two months, and nobody notices the drift until an auditor does. The second is letting junctions become rubber stamps. Junction holders approve the executor's recommendation every time, so the junction exists on paper only. Overrides near zero for months are a warning, not a success. The third is never using the switch-off, even when it is clearly needed, because switching off feels like admitting failure. The fourth is widening the scope without recording it, until nobody can say what the loop is allowed to do.



Like a checkpoint guard who waves every car through: *the barrier exists on paper, and months with no stops are a warning sign, not a success.*

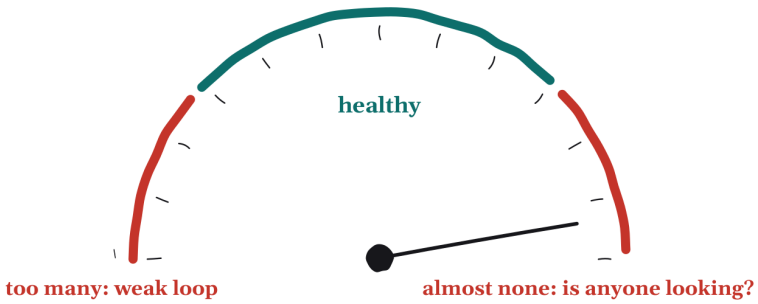
What the foundations add

Core is the closed loop running. Norbert Wiener described it in 1948: act, sense the result, compare it with the goal, adjust. Every outcome is graded

against the standard, the grades are read at the monthly review, and the review changes rules, limits or context. A loop that acts without that cycle is running open, however well it was designed.

Drift is what open loops miss. The world the loop reads keeps moving: suppliers change formats, volumes shift, a new regulation adds a check. A learned model trained on the old distribution goes on answering confidently in the new one. Drift checks compare today's cases with the ones the loop was proven on.

The engineer Lisanne Bainbridge named the ironies of automation: the more reliable the automation, the less practised the people who must step in when it fails. Override rates are therefore a health signal in both directions. Too many overrides mean the loop is weak. Almost none, for months, can mean people have stopped looking.



Like an engine gauge red at both ends: *too many overrides mean a weak loop, and almost none for months can mean nobody is looking.*

At the manufacturer

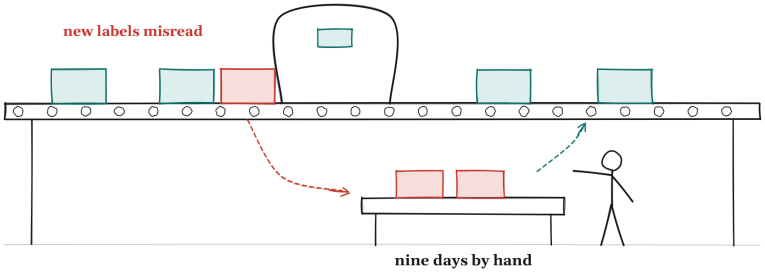
The full scope went live in month eight. Non-order invoices under €5,000 were added, with the cost-centre coding as a bounded decision. New suppliers came into scope after three months of history instead of six. Invoices over €50,000 stayed with the controller, and bank-detail changes with treasury.

In its third month in Core, one of the company's largest suppliers changed the layout of its invoices without warning. The executor began misreading the order reference on about 1.6% of that supplier's invoices. The grader flagged a drop in quality for that supplier within two days. The head of accounts payable routed all of that supplier's invoices to the clerks for nine days while the reading was corrected and re-tested. Then they were returned to the loop. The operating review that month recorded the event, the response and one new

rule: any supplier whose quality drops below the standard for two days in a row is routed to people automatically.

The monthly reviews also tracked junction overrides. The controller overrode the executor's evidence on about one large invoice in twenty, usually because of information the executor could not see, such as a phone call with the supplier. The review treated that rate as healthy and asked for those reasons to be logged, so the context node could be improved.

The head of accounts payable signed the switch-off plan and each operating review. After the third review, all twelve nodes were signed, and the loop stood at G4.



Like a grocer whose scanner misreads one supplier's new labels: *those crates go to the hand-check bench for nine days, then back onto the belt.*

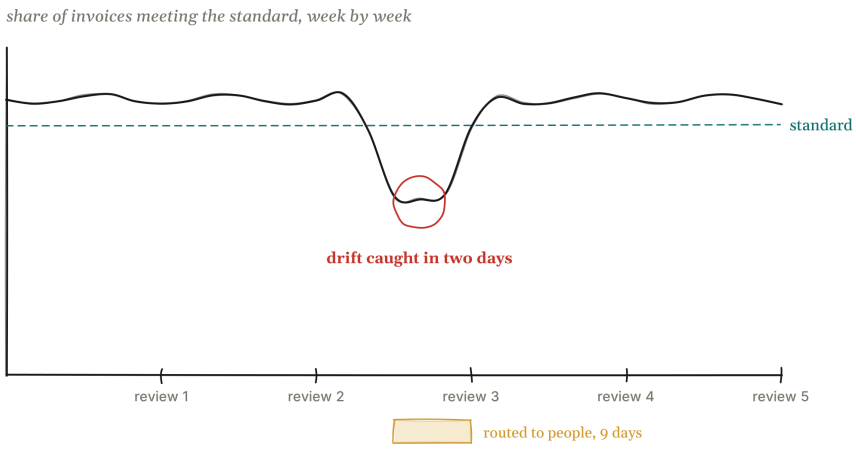


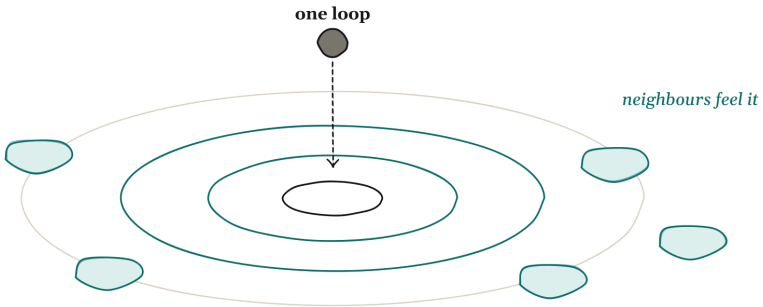
FIGURE 17 Step 9, Core. Quality is graded on every outcome. When a supplier changed its invoice layout, quality for that supplier dipped, the grader caught it, and its invoices went to people for nine days.

XVIII

Step 10: Scale

The step at a glance	
Question	What does this loop change across the map, and which loop comes next?
Starts when	At least one loop has passed G3, and usually G4
Signed	The map review and the role change plan
Signer	The executive sponsor
Exit gate	The map updated, the next loops chosen, the changes to roles agreed; the next Read begins
Typical duration	A quarterly review, about two weeks of preparation

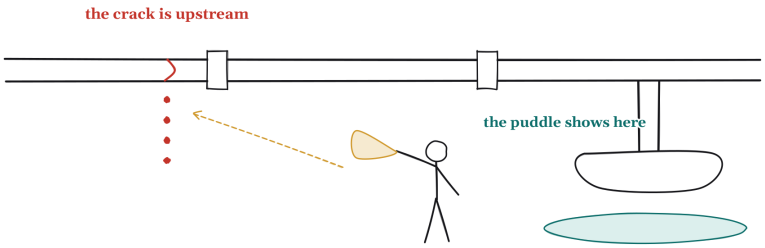
The last step looks up from the single loop to the whole map. Embedding one loop changes things around it. Work that was waiting for that loop now arrives sooner. Problems that loop used to absorb now show up at its edges. People have moved. And the company has learned how long each step really takes. Scale turns those changes into decisions about what comes next.



Like a stone dropped into a lily pond: *the ripples spread from the one spot and reach every neighbouring pad, changing what sits around it.*

Reading the edges

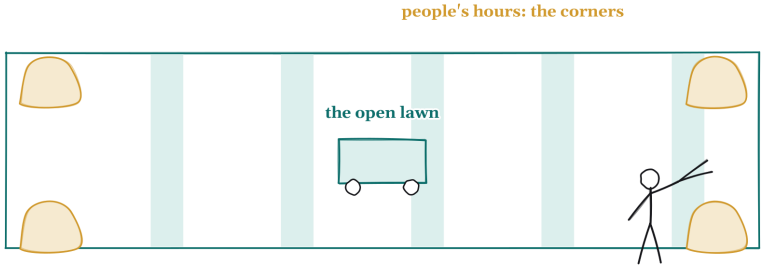
The best guide to the next loop is the handoffs of the last one. A loop running in the core makes its neighbours visible. The exceptions it cannot clear usually point to a loop upstream that sends it incomplete work. The waiting it can no longer explain usually sits in a loop downstream. Reading the exception queue by cause is often the fastest way to see which loop should go next.



Like a plumber tracing a puddle: *the water pools under the sink, but following the drips back points to the cracked pipe upstream.*

Re-reading the distribution

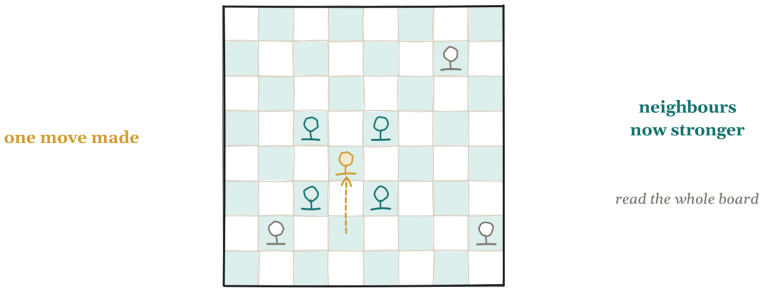
Scale also returns to the distribution. Once an executor runs the body of a loop, the people′s time concentrates in the tail. That changes what the company should improve next. It may be the junction evidence, so tail decisions are faster. It may be a new loop for a class of tail cases. Or it may be a decision that the tail is exactly where people should stay.



Like a garden after a robot mower takes the open lawn: *the gardener′s hours move to the edges and corners, so better shears now matter most.*

The map review

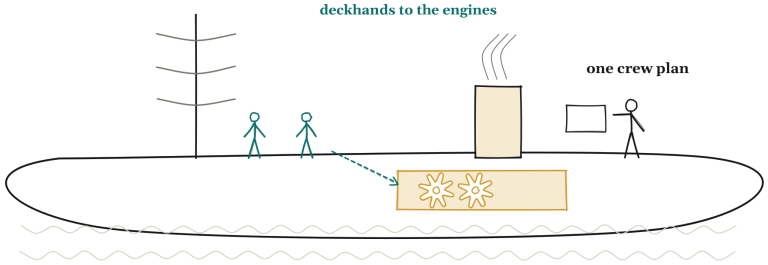
The executive sponsor holds a map review, usually each quarter. It updates every loop′s readiness, since the work on one loop often raises the readiness of its neighbours. It records the state of each lifecycle in progress, step by step. And it chooses the next loops to Read. The review is signed by the sponsor. It is the same document that began the lifecycle at Step 1, one cycle later.



Like a chess player studying the whole board after each move: *one move changes its neighbours' strength, so the next move is chosen from the new position.*

The role change plan

Every loop that reaches Own changes some people's work. Across a map, those changes add up, and they need a plan larger than any single loop. The role change plan records which roles are growing, such as junction holders, exception owners, graders and business engineers. It records which are shrinking, where people are moving, and how expertise will be rebuilt now that routine cases no longer teach it. It is signed by the sponsor, because it commits the company, not just one team.

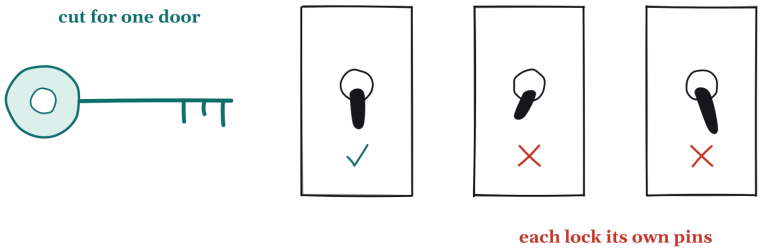


Like a ship refitted from sail to engine: *deckhands become engineers, and the captain plans the whole crew's moves, not one mast at a time.*

How it fails

The first failure is never coming back to the map. The first loop succeeds, the team moves straight to the next project someone suggests, and the company ends up with a scattering of embedded loops that do not connect. The second is scaling by copying. The design of one loop is applied to another without running the lifecycle, on the theory that they are similar. They rarely are. Each loop's rules, decisions and distribution are its own. The third is leaving the role

changes to each loop, so that across the company nobody sees the whole shape of the change to people's work.

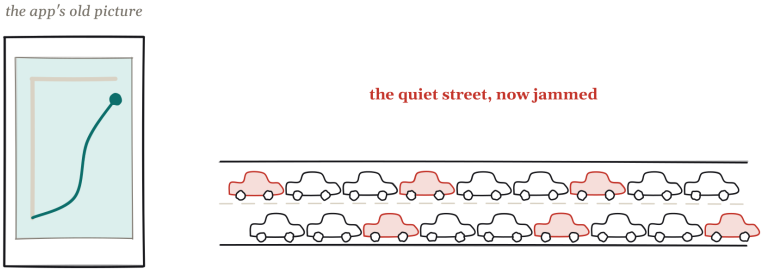


Like cutting one key and trying it in every door: *the first lock opens, but each of the others has pins of its own.*

What the foundations add

A running loop changes the distribution it reads. The financier George Soros called this reflexivity: actions based on a picture of the world alter the world the picture describes. Once the body of invoices runs alone, suppliers learn which invoices clear fastest, the tail shifts, and some old exceptions disappear while new ones appear. The reading made at Step 1 is now out of date, partly because the loop succeeded.

So Scale re-reads before it moves on. Hofstadter called a system that models itself and acts through the model a strange loop. A company that embeds AI becomes one: it builds a model of its work, acts through it, and the action changes the work. The loop stays honest only if the model is redrawn from the real work each time, not from the previous model. **That is what turns one embedded loop into a capacity to embed.**



Like a traffic app that sends everyone down the quiet street: *the street fills, so the picture must be redrawn from the real traffic each time.*

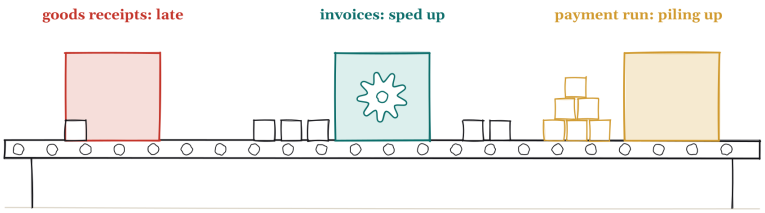
At the manufacturer

The first map review came in month twelve. Reading the invoice loop's exception queue by cause showed that missing goods receipts caused 38% of the exceptions that remained. The receiving teams at the three plants still confirmed deliveries late, or not at all. The goods receipt loop, already marked on the map, moved to the top. It entered Read the following month.

The payment run was chosen second. With invoices ready in three days instead of sixteen, the weekly payment run had become the constraint between an approved invoice and a paid supplier.

Re-reading the invoice distribution showed where people's time now went. The 3% of invoices over €50,000 took more than half of the remaining hours, mostly at the controller's junction. The review asked for better evidence on large invoices, starting with contract terms and delivery history in one view.

The role change plan recorded the moves from the invoice loop and planned the next ones. The receiving teams' work would change with the goods receipt loop. Two of the former clerks, now in supplier onboarding, would help map it. The CFO signed the map review and the plan, and the lifecycle began again at Read.



Like an assembly line after one station is sped up: *work now piles up at the next station, and the late parts from upstream become obvious.*

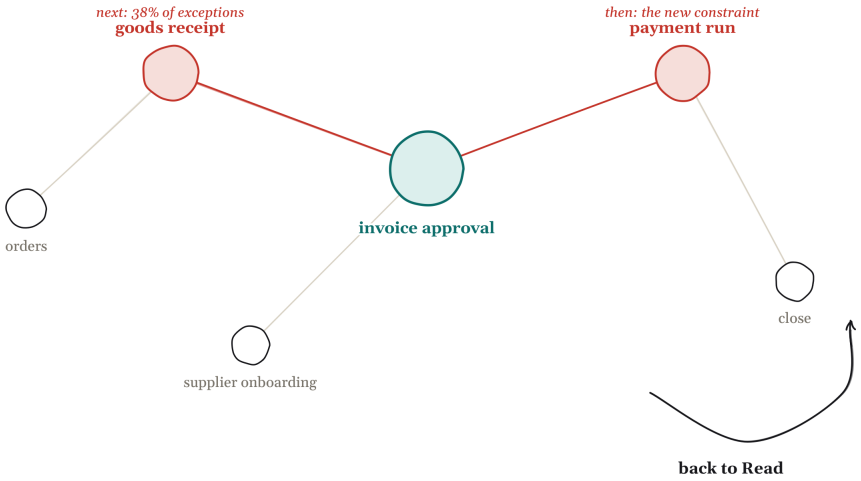


FIGURE 18 Step 10, Scale. The embedded loop lights up its neighbours: missing goods receipts point upstream, the payment run waits downstream, and the map review sends the lifecycle back to Read.

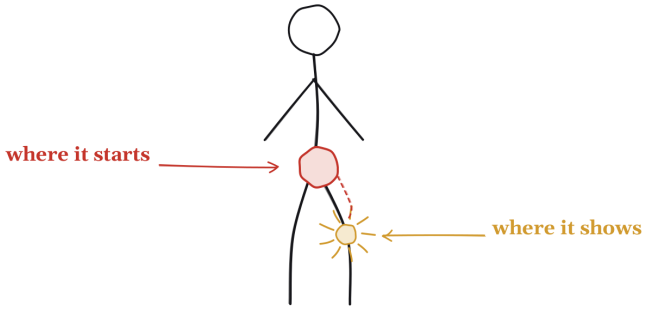
PART SIX

Across the Lifecycle

XIX

How Lifecycles Break

Each step chapter listed its own failures. This chapter steps back and looks at the patterns, because most stalled programmes fail in one of a small number of ways. Each failure has a step where it could have been prevented, and that step is rarely the one where the failure shows.

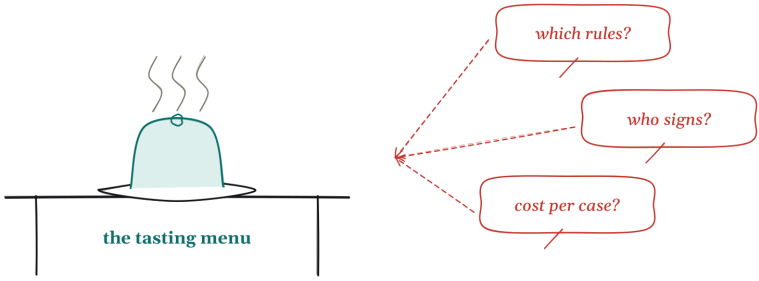


Like referred pain: *the knee hurts, but the cause sits in the hip, so each failure must be traced back to the step that could have prevented it.*

Starting in the middle

The most common pattern is a programme that begins at Step 6. A pilot built in a few weeks is put beside the real process, the results look good, and the team prepares to switch it on. Then the questions arrive. Which rules is it following? Who approves the large cases? What happens to the cases it cannot handle? What does it cost per accepted result, against what baseline? None was asked earlier, so all are asked at once, at the point where every answer delays launch.

The prevention is in Steps 1 to 5. A pilot is valuable evidence that the executor can do the task. It belongs inside Shadow, after the work of Read, Map, Strip, Sort and Design has been done.

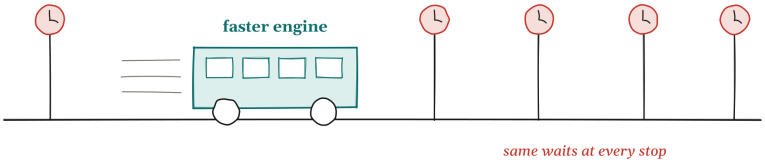


Like a restaurant that serves its tasting menu before getting its licences: *the food wins praise, then every question arrives on opening night and each one delays it.*

Automating the workaround

The second pattern is a loop that works and saves little. The executor runs every step of the old map faster. The waits between steps stay where they were, and the elapsed time barely moves. Often the executor is diligently re-keying data between two systems that could have been connected, or completing a review that guarded against an error nobody makes any more.

The prevention is Step 3. Strip before you sort, and sort before you design.



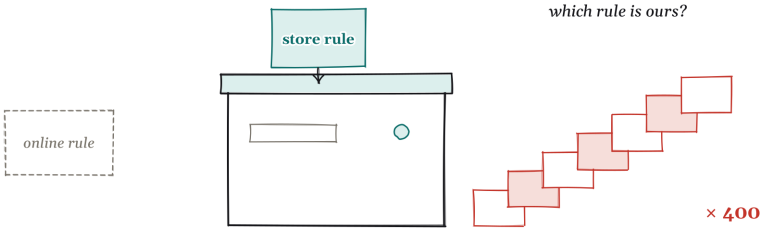
Like a faster engine in a bus that still calls at every old stop: *the drive between stops shrinks, but the waits remain and the journey barely shortens.*

The incident that was a decision

The third pattern is an incident that turns out not to be a technical failure. The executor paid a disputed invoice, refunded a customer outside policy, or accepted a claim that one manager would have rejected. Investigation shows that it followed one of two rules that the company had been applying inconsistently for years. The executor did not create the disagreement. It made the disagreement visible, at scale, in the ledger.

The prevention is Step 4, and specifically the contested-decision log. Every contested decision found and settled before Design is an incident that does not happen.

A composite case shows the pattern. A retailer embedded its returns-and-refunds loop. Within a month, the executor had approved several hundred refunds on items bought outside the return window, from customers with long purchase histories. The investigation found two rules in use. Store managers had always made goodwill exceptions for loyal customers, while the online team never had. The executor had learned from the store data. Nobody had ever decided which rule was the company's, and the refunds were the first time anyone had to.



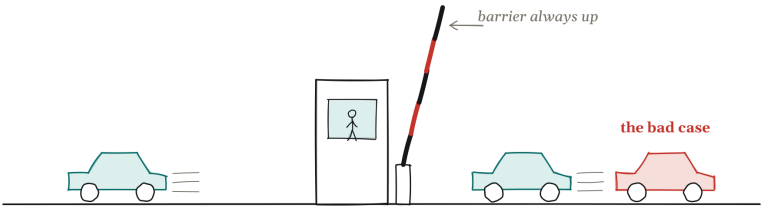
Like a photocopier fed one shop's note: *an exception made quietly for years comes out by the hundred, and someone must finally decide the company's rule.*

Junctions that do not decide

The fourth pattern appears months into Core. Junction holders approve the executor's recommendation almost every time, often without opening the evidence. The junction exists on the canvas, but nobody is really deciding. When a bad case arrives, it passes the junction as easily as the good ones.

The prevention is partly in Step 5, which designs what evidence a junction holder sees. It is partly in Step 9, whose operating review tracks override rates and treats a rate near zero as a warning.

A composite bank met this pattern in its customer onboarding loop. Compliance analysts held the junction for high-risk applicants. After six months, they were approving 99.6% of the executor's recommendations, often in under a minute. A sample review found that the evidence screen showed the executor's conclusion first and the source documents three clicks away. The screen was redesigned to show the documents first and the recommendation last. Override rates rose to a level the head of compliance considered credible.



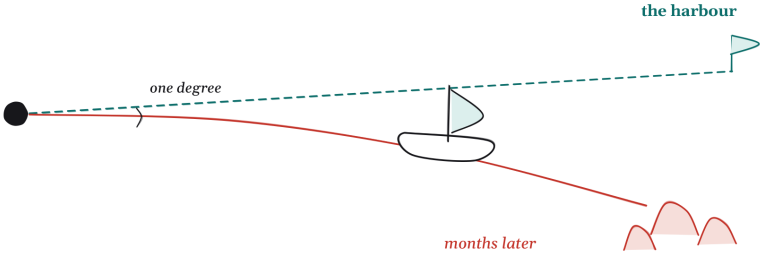
Like a toll barrier stuck in the raised position: *the booth is staffed, yet every car drives straight through, the bad one as easily as the rest.*

Drift nobody watched

The fifth pattern is slow decline. Quality falls by a point or two a month as inputs change, until a customer, a regulator or an auditor notices. The programme team has moved on. The operating reviews became status meetings and then stopped.

The prevention is the standard from Step 5, which grades every outcome, and the operating review from Step 9, which reads the grades body and tail separately every month.

In a composite insurer, a claims intake loop ran well for a year. The model provider then released an update, and the loop began classifying a small share of water-damage claims as maintenance issues. The overall quality figure barely moved, because the share was small. The tail did move: larger claims were being routed wrongly. The insurer had stopped reading body and tail separately after the first quarter. A broker's complaint found the problem, four months late.

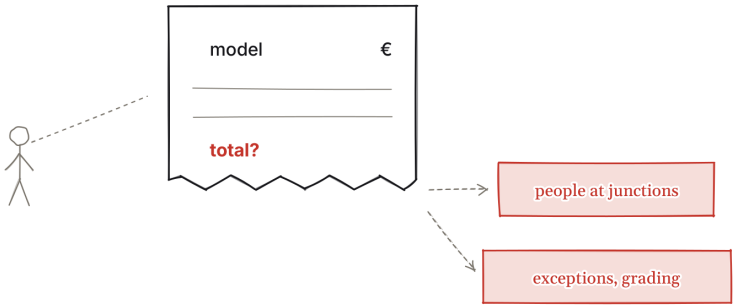


Like a ship one degree off course: *nothing looks wrong on any single day, and only regular checks of position against the chart catch it before the rocks.*

Savings that were never measured

The sixth pattern is a loop that runs well and still cannot defend its budget. When finance asks what it saved, the answer compares an estimate of the old cost with a partial count of the new one. The model cost is counted, but the people at junctions, the exception queue and the grading are not. Finance does not believe the number, and it is right not to.

The prevention is the baseline at Step 2 and the economics at Step 8, measured the same way, signed by finance.

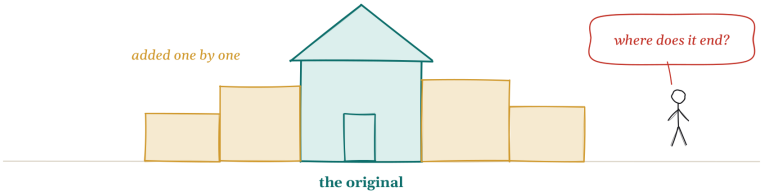


Like a receipt with its bottom torn off: *the model's charge is printed, but people at the junctions, exceptions and grading are missing, so finance rejects the total.*

Scope that crept

The seventh pattern is a loop whose limits nobody can state. It began with small, order-backed invoices. Over a year, case types, value bands and counterparties were added one at a time, each by an informal agreement. When an auditor asks what the executor is allowed to do, the answer takes a week to assemble.

The prevention is the gate review at Step 7 and the rule at Step 9 that every widening of scope is recorded and agreed by the control owner.

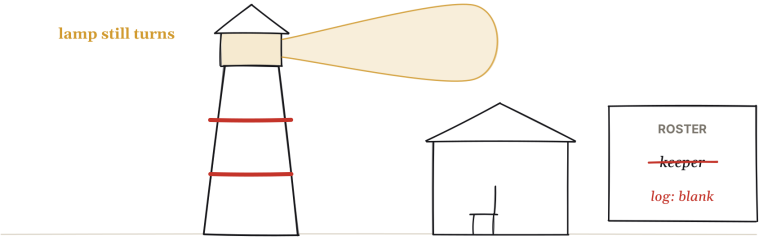


Like a cottage extended one room at a time by handshake: *years later nobody can show the inspector where the approved building ends.*

A loop with no owner

The last pattern is the quietest. The loop runs, but the person who owned it has moved jobs, and nobody took the role formally. Exceptions age. Reviews stop. The switch-off plan names someone who has left.

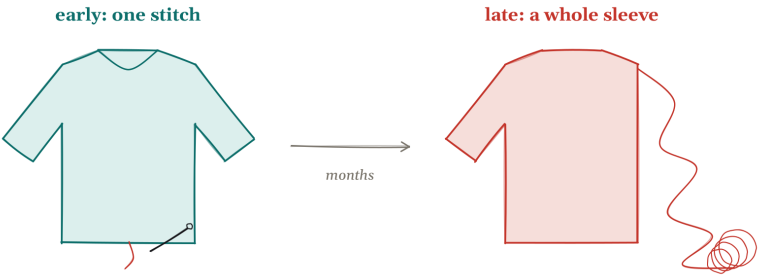
The prevention is Step 8. The charter names people, not roles. And the operating review at Step 9 should check, every month, that every named person is still in post.



Like a lighthouse whose keeper moved away: *the lamp still turns each night, but nobody keeps the log, and the roster still names someone long gone.*

Where the failures cluster

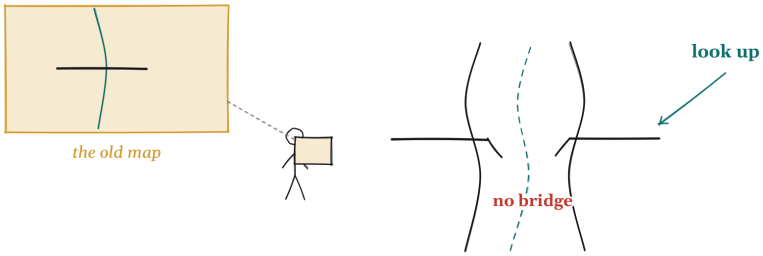
Put these patterns on the lifecycle and a shape appears. Most failures show up late, in Shadow, Bound or Core, where they are expensive. Most are prevented early, in Map, Strip, Sort and Design, where they are cheap. **That asymmetry is the strongest argument for running the steps in order.** The early steps feel slow because their value is in the failures that never arrive.



Like a loose thread on a jumper: *one stitch fixes it early, and left alone it unravels a whole sleeve later, so the cheap early steps come first.*

What the failures have in common

Read through the foundations, the eight patterns reduce to two. In some, the loop runs open: nobody senses its results, so drift, creeping scope and unmeasured savings go unnoticed. In the others, the company's model of its own work stops mirroring the work. The workaround is automated as if it were a rule, the incident turns out to be a decision nobody made, the owner on the charter has left. The cure in both cases is the same. Keep the loop closed, and redraw the model from the real work rather than from the last version of the model.



Like a walker reading an old map instead of the ground: *look up to see where you are, and redraw the map from the path itself.*

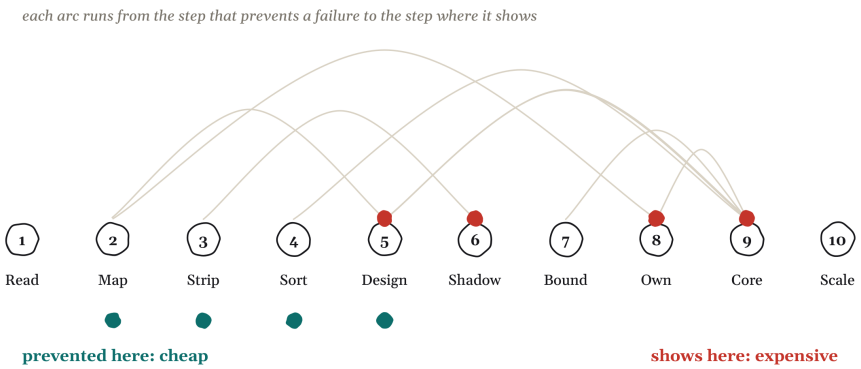
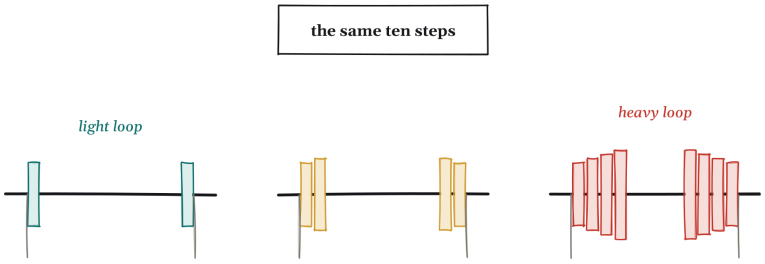


FIGURE 19 Where failures show and where they are prevented. Most failures appear late, in Shadow, Bound or Core, where they are expensive. Most are prevented early, where they are cheap.

XX

Adapting the Lifecycle

The ten steps hold for every loop. How long each takes, and what it emphasises, depends on the loop. This chapter covers the six adaptations that come up most often. None of them removes a step or a signature. They change the weight each step carries.

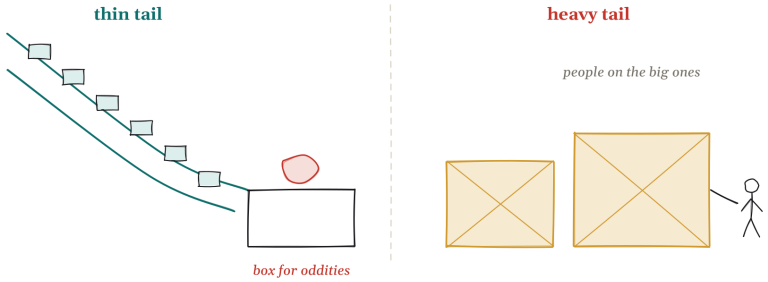


Like one set of exercises for every athlete: *none is skipped, but the weight on each bar is set for the person lifting it.*

By the shape of the distribution

In a loop whose cases are mostly alike, such as standard timesheet approvals or routine address changes, the tail is thin. Shadow can be shorter, because agreement on the body tells you most of what you need. Bound can cover most of the volume from the start, and junctions are few. The risk in these loops is complacency: a rare odd case arrives months later and finds no route to a person. The exception queue matters more than the junctions.

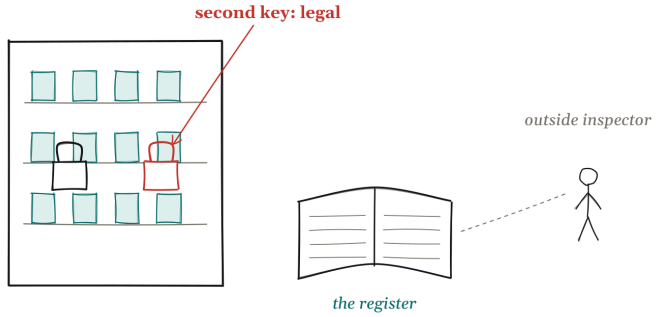
In a loop with a heavy tail, such as large claims, complex contracts or high-value credit, most of the value and risk sit in a few cases. Shadow must run long enough to see enough of them, often two or three months. Bound covers a smaller share of the value, even if it covers most of the volume. The junction evidence deserves the most design effort, because that is where people will spend their time. In these loops the executor's biggest contribution is often preparing the tail cases for people, not deciding them.



Like two post rooms: *one sorts near-identical letters down a chute with a box for oddities, the other gets a few heavy crates that need people.*

Regulated loops

Some loops fall within rules that govern automated decisions. Examples include hiring and managing people, credit and insurance pricing, access to essential services, and anything regulated as high risk under laws such as the European Union's AI Act. For these, two additions are needed. Legal or compliance joins Sort, to confirm which decisions the law requires a person to take or review. And it joins the gate review at Bound, to confirm that oversight, logging and monitoring meet the legal standard, not just the company's own. The junction register and the switch-off plan become legal evidence. Write them so that someone outside the company could follow them.

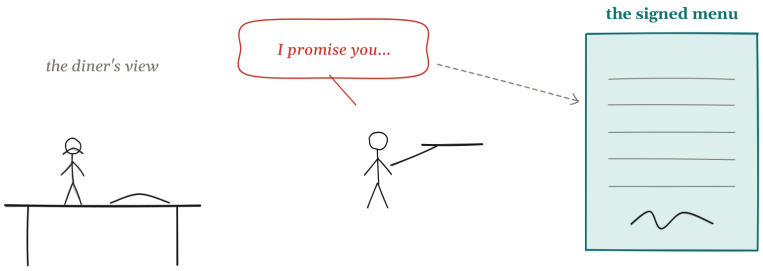


Like a pharmacy's controlled-medicines cabinet: *a second key holder from outside the counter, and a register written so an inspector can follow every entry.*

Customer-facing loops

When the receiver of a loop's outcome is a customer, the standard has to include the customer's view, not only internal correctness. A support reply can be accurate and still fail the customer. Complaints, repeat contacts and

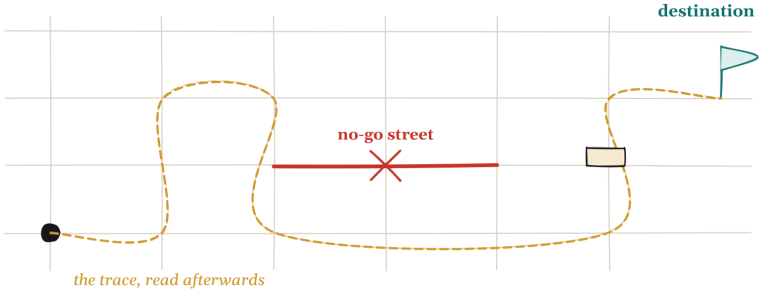
reopened cases become part of the grading. Junctions often sit where the executor would make a promise on the company's behalf: a refund, a delivery date, an exception to policy. A company is generally bound by what its agents tell customers, so the executor's promises must stay inside signed limits.



Like a waiter: *a correct dish can still leave the diner unhappy, and whatever the waiter promises, the restaurant must honour, so promises stay within the signed menu.*

Loops whose steps are generated

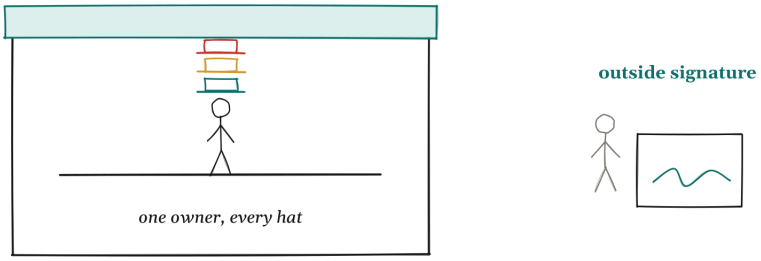
In some loops the executor does not follow a set sequence at all. It plans its own steps for each case: research for a proposal, investigation of an incident, drafting an analysis. Mapping still matters, because it records what the work achieves today and what it costs. But the design moves up. The steps become a trace of what the executor did, not a specification. The canvas carries the weight: outcome, rules, limits on actions, junctions and the standard. Grading every outcome becomes even more important, because there is no fixed path to audit.



Like a taxi with no fixed route: *you set the destination, the fare limit and the streets it may not take, then read its route trace afterwards.*

Small companies

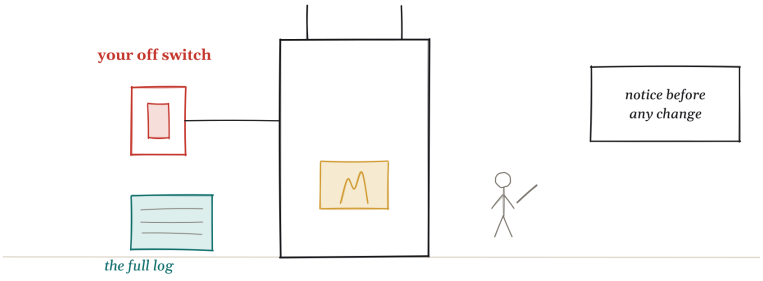
A company of a few hundred people may have one person who is sponsor, process owner and control owner at once. The lifecycle still applies, and the steps are faster because fewer people must agree. One separation should survive even in a small company. The person who answers for the loop's results should not be the only person who signs its limits at Bound. If nobody else is available, an outside accountant or adviser can take the control owner's role for that one review.



Like a corner shop whose owner wears every hat: *the work is quicker, but an outside accountant still signs the limits the owner works within.*

Loops run by a vendor

Some executors are bought as a service: the vendor's platform, the vendor's model, sometimes the vendor's staff at the junctions. The lifecycle does not change, but the contract has to make it possible. The company needs the full log of every case, the ability to switch the loop off, access for its own grader or auditor, and notice before the vendor changes the model. It also needs the baseline and economics measured on the company's own numbers, not the vendor's case studies. A vendor that will not agree to these terms is selling a pilot, not an embedded loop.



Like a contractor running your boiler: *insist on the full service log, your own off switch, access for your inspector and notice before any part is changed.*

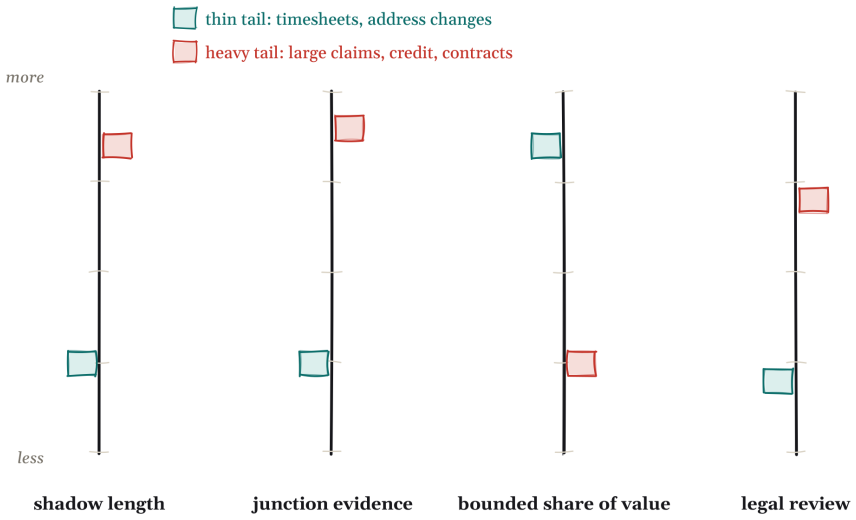
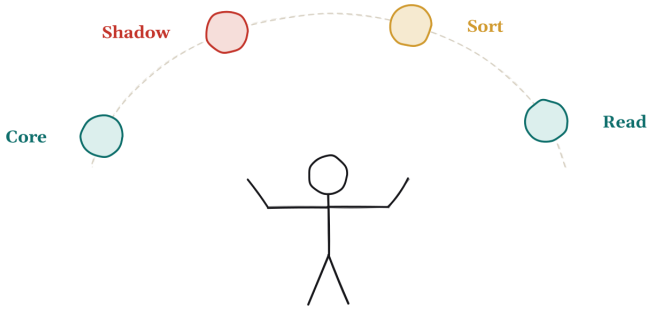


FIGURE 20 Adapting the lifecycle. The steps and signatures stay the same for every loop; what moves is the weight each one carries. A thin-tailed loop and a heavy-tailed loop set the same four controls very differently.

XXI

Running Many Loops at Once

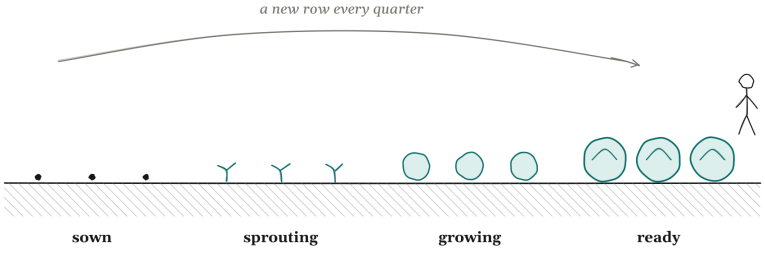
The lifecycle has been described for one loop. A company that takes embedding seriously soon runs several at once, each at a different step. Running many lifecycles is a discipline of its own, and it has a few rules.



Like a juggler: *several balls in the air at once, each at a different point of its arc, which is a skill of its own.*

Stagger the steps

Loops should not move through the lifecycle in lockstep. The steps draw on different people. Read and Scale draw on the executive sponsor. Map and Strip draw on operators. Sort draws on process owners and leaders. Shadow and Bound draw on graders and control owners. Own draws on finance. Staggering the loops spreads the load. A sensible pattern is to start a new Read each quarter, so that at any time one loop is being mapped, one redesigned, one proven and one running.

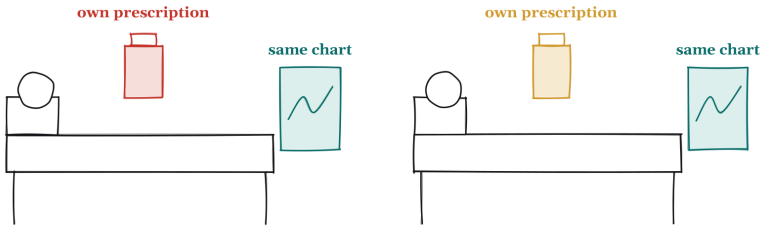


Like sowing a new row of lettuce each month: *one row sprouts, one grows, one is picked, so the gardener's work and the harvest stay steady.*

Share the instruments, not the designs

Every loop uses the same instruments. These are the map, the canvas, the registers, the gate review record, the charter, the economics sheet and the operating review. Shared instruments let leaders compare loops, let auditors find what they need, and let people move between loops without relearning the paperwork.

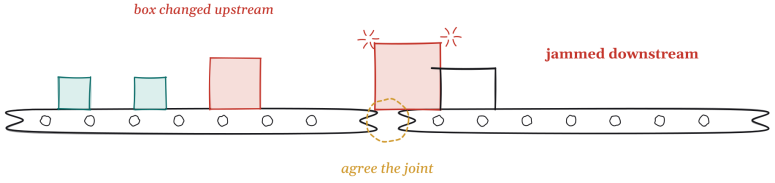
What should not be shared is the design. Each loop's rules, decisions, junctions and distribution are its own. Two loops that look alike on the map often turn out to differ on the one decision that matters.



Like a hospital ward: *every patient gets the same chart and the same thermometer, but each gets a prescription written for their own case.*

Watch the handoffs between loops

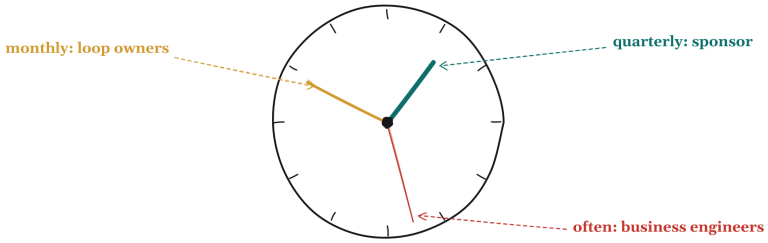
As more loops reach the core, the edges between them matter more than their insides. A loop that sends work to another must agree what it sends, in what state, and who resolves a mismatch. Those agreements live in the handoffs node of each canvas. They should be reviewed whenever either loop changes, because a small change upstream can fill a downstream exception queue overnight.



Like two conveyor belts joined end to end: *a small change to the boxes upstream jams the next machine overnight, so the joint is agreed and rechecked.*

One operating rhythm

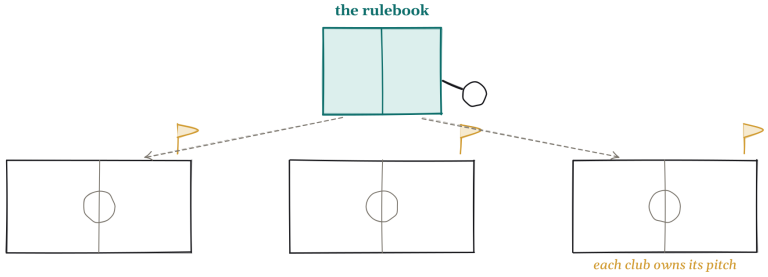
A company running many loops benefits from a single rhythm. Each loop owner holds the monthly operating review for their loop. The sponsor holds the quarterly map review, reading the state of every lifecycle and every loop in the core. The business engineers who run the lifecycles meet more often, compare notes, and keep the instruments consistent. The rhythm is simple, but without it each loop drifts into its own habits, and the company loses the ability to see its embedding as a whole.



Like a clock's three hands: *each turns at its own pace, the quick one, the monthly one, the quarterly one, but all turn on one face.*

The business engineering function

At some scale, usually around the fifth or sixth loop, companies give the lifecycle a home. It can be a small team in transformation, finance or operations. Its job is to run lifecycles, train loop owners, keep the instruments, and hold the map. It does not own any loop; owners do. It owns the method, and it is accountable for the method being followed.



Like a referees' association: *it owns no club and plays no match, but it trains the referees, keeps the rulebook and answers for how it is applied.*

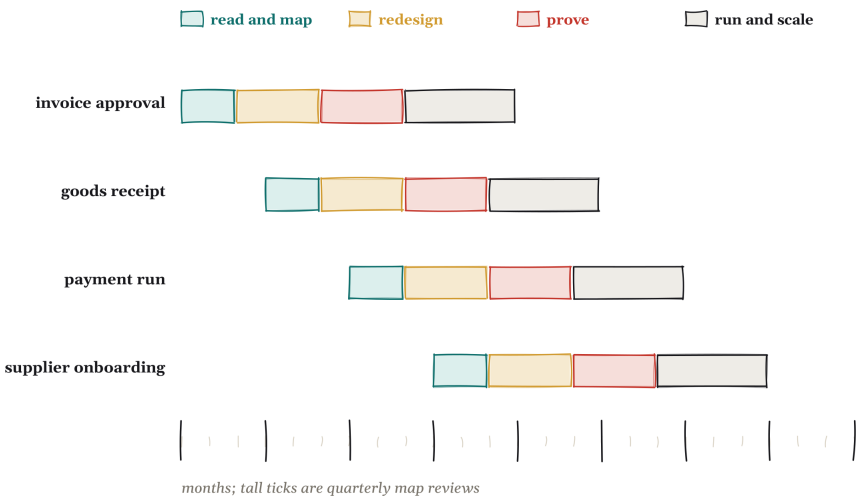
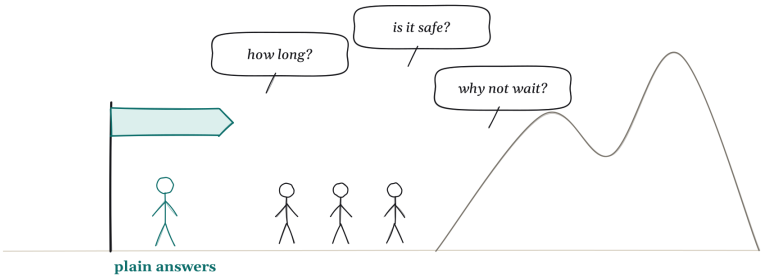


FIGURE 21 Running many loops at once. A new Read starts each quarter, so at any time one loop is being mapped, one redesigned, one proven and one running, all on one operating rhythm.

XXII

Questions Leaders Ask

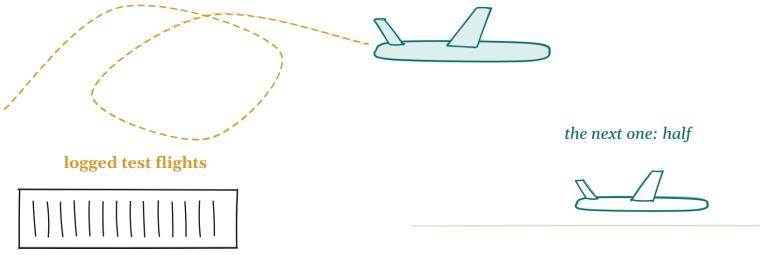
Executives meeting the lifecycle for the first time tend to raise the same handful of questions. Here are the most common, with direct answers.



Like a mountain guide's briefing at the trailhead: *every group asks the same few questions, and a good guide answers each one plainly before setting off.*

"Seven to ten months is too slow."

It is slower than a pilot and faster than most programmes. A pilot reaches a demo in weeks, but the surveys suggest most never reach production at all. The lifecycle's time is mostly spent collecting evidence: weeks of shadow running, weeks of measured economics. Remove it and the evidence is replaced by hope. The second loop takes roughly half as long, and a company running several loops at once sees progress every month, not every year.

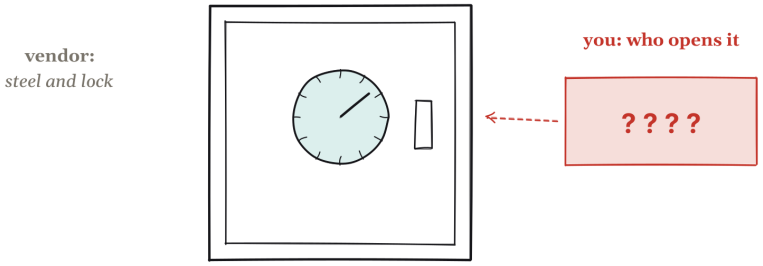


Like test flights for a new aircraft: *weeks of logged flying before passengers board, and the next plane of the same type needs about half as long.*

"Our vendor says its product does all of this."

A vendor can supply the model, the platform and often the grading. It cannot decide which of your rules is right when two contradict, who in your company

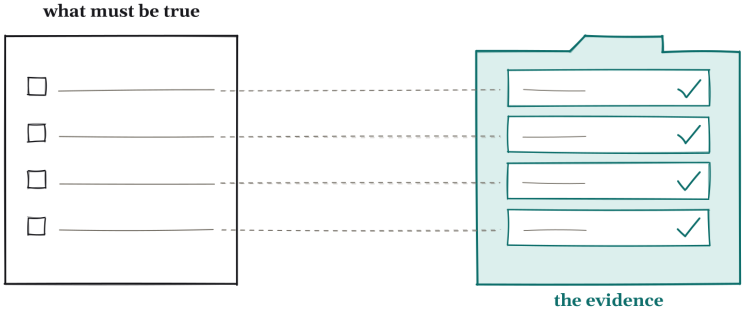
releases a large payment, or whether to pay a disputed invoice to protect supply. Those decisions are the substance of Steps 3 to 8. Ask the vendor which of your signed documents its product produces. The honest answer is none, though a good product makes several of them easier to keep.



Like buying a first-rate safe: *the maker supplies the steel and the lock, but only you can decide who holds the combination and what goes inside.*

"We already have an AI governance framework."

Good. Governance frameworks say what must be true: oversight, risk assessment, monitoring, accountability. The lifecycle is how a particular loop comes to make those things true, step by step, with records. Appendix E maps the two. Companies with a framework usually find the lifecycle supplies the evidence their framework asks for.

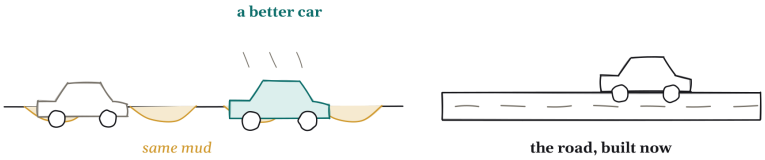


Like an exam syllabus and a coursework folder: *the syllabus says what must be shown, and the folder is the evidence, piece by piece, that it was.*

"Models are improving so fast. Why not wait?"

Better models make the executor more capable. They do not make the company's rules consistent, settle its contested decisions, measure its baselines or name its junction holders. That work is the same whichever model is used, and it takes the same time. A company that waits for better models will meet

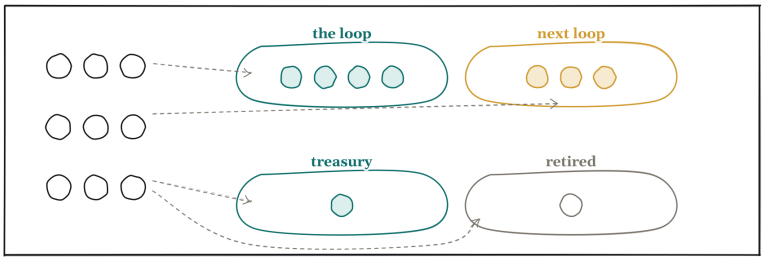
the same middle with a better model. A company that does the work now can swap in better models as they arrive, and its grading will show whether they help.



Like waiting for a faster car before building the road: *the new car sinks in the same mud, while a road built now carries every later car.*

"Will this cost jobs?"

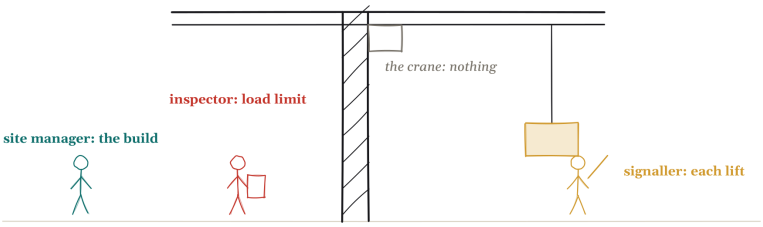
It changes jobs, and in some loops it reduces the number of people doing routine work. At the manufacturer, nine clerks became four on the loop, three moved to another loop, one moved to treasury and one retirement was not replaced. The lifecycle does not decide headcount. It insists that changes to people's work are agreed openly at Own and planned across the map at Scale, and that the company plans how people will learn the judgement the junctions need. Programmes that hide these questions lose the trust of the people whose knowledge they need at Map and Sort.



Like a coach redrawing the tactics board in front of the squad: *of nine players, four keep their places, three move positions, one transfers, one retires.*

"Who is accountable if the executor gets it wrong?"

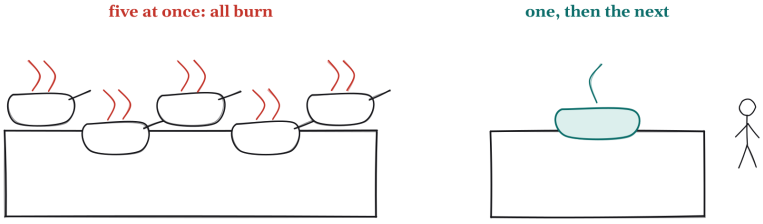
The loop owner answers for the outcome, as they would for any team's work. The junction holder answers for the decisions taken at the junction. The control owner answers for the limits signed at Bound. The executor is never accountable, because it cannot be. The lifecycle exists largely to make this answer clear before the first live case, rather than after the first incident.



Like a building-site crane: *the site manager answers for the build, the signaller for each lift, the inspector for the load limit, the crane for nothing.*

"Can we run several loops in parallel from the start?"

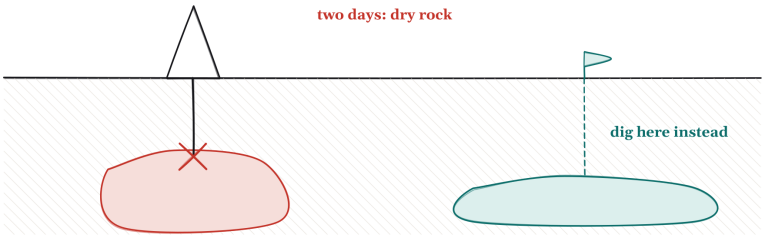
Not usually. The first loop teaches the company the steps, the instruments and how long things really take. Running five first loops at once multiplies the learning cost and usually stalls all five at Sort, where the same leaders are needed for every contested decision. Start one. Start the second once the first passes Shadow. Then start one a quarter.



Like a new cook: *five pans at once all burn while waiting for the same hands, so master one dish, then add a pan at a time.*

"What if a step shows we picked the wrong loop?"

Stop and go back to the map. Finding at Map that a loop's baseline is already excellent, or at Sort that its decisions are mostly contested, is a success of the lifecycle. Two weeks spent learning that is far cheaper than six months spent embedding the wrong loop.

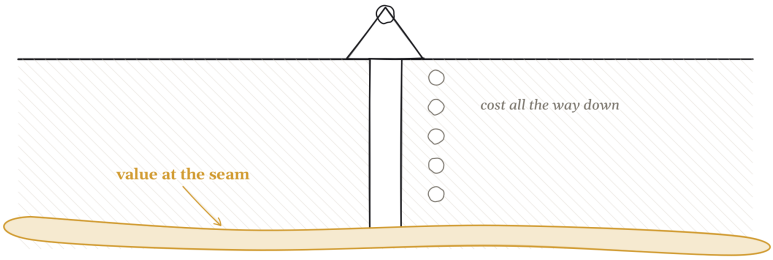


Like a test bore before digging a well: *hitting dry rock after two days is a success, far cheaper than months spent digging in the wrong place.*

XXIII

What Finishing Is Worth

It is worth being precise about where the value of embedding appears, because it explains both why the middle is so expensive and why the lifecycle pays for itself.

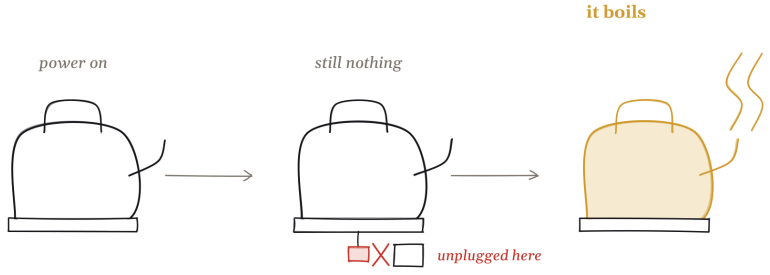


Like a mine shaft: *money is spent all the way down, but the ore lies only at the seam, so know exactly how deep it is.*

Value arrives late

A pilot produces evidence, not value. Shadow produces more evidence and some valuable findings, such as rules nobody knew were inconsistent. But it saves nothing, because people are still doing all the work. The first real savings arrive at Bound, on the body of the distribution. They grow at Core, as the scope widens and the old process is retired.

So the value curve of an embedding programme is flat for months and then rises steeply. A programme that stops anywhere before Bound has spent most of its cost and captured almost none of its value. That is the economic shape of the middle, and it is why so many surveys find so little return. Most programmes are spending money in the flat part of the curve.



Like a kettle: *it draws power for minutes with nothing to show, then boils, and switching off just before the boil buys only lukewarm water.*

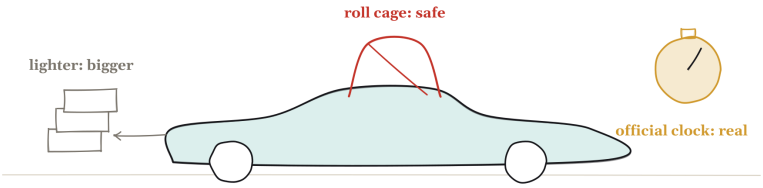
What the steps protect

The early steps look like cost, but they protect the later value in three ways.

They make the savings bigger. Strip removes waits and steps that an executor would otherwise have run faster but kept. At the manufacturer, most of the drop in elapsed time came from the six struck steps, not from the executor's speed.

They make the savings real. A measured baseline and a fully loaded economics sheet mean finance believes the number, so the loop keeps its budget and the next loop gets one.

They make the savings safe. A contested decision found at Sort, a junction placed at Design and a switch-off tested at Bound each prevent a single incident. One such incident can cost more than the loop saves in a year, and can stop the whole programme.

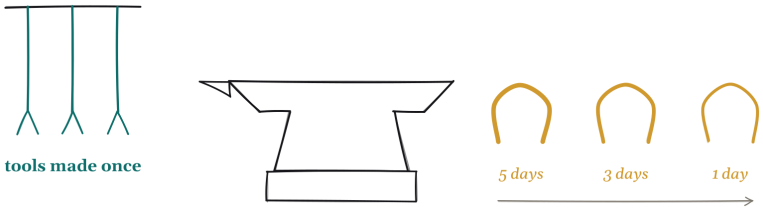


Like preparing a race car: *stripping weight makes it faster, the official clock makes the lap count, and the roll cage stops one crash ending the season.*

The value of the second loop

The first loop through the lifecycle is the most expensive. The map is drawn for the first time, the instruments are new, the people are learning the steps. The second loop is cheaper and faster. By the fourth or fifth, the cost of each lifecycle falls sharply, and the company has something more valuable than any single loop: the capacity to embed.

That capacity is what separates the few companies that report large returns from AI from the many that do not. They did not find better models. They built the ability to move loop after loop through the middle.

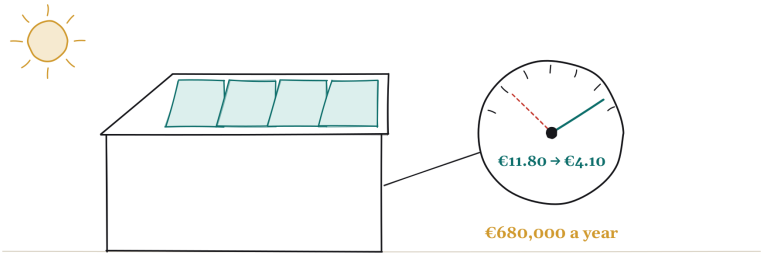


Like a new blacksmith: *the first horseshoe takes days because the tools are made too; by the fifth, shoes come fast and the forge is the asset.*

At the manufacturer

The invoice loop cost about four months of effort before it saved anything. The cost included the time of the clerks and the controller, the business engineer, the platform and the shadow period. At Bound, savings began on three quarters of invoices. At Core, the measured cost per accepted invoice of €4.10 against €11.80 saved the company about €680,000 a year on invoice processing. Paying suppliers on time also let it capture early-payment discounts it had been missing, and it had fewer late-payment disputes.

The CFO's own summary at the first map review was simpler. The invoice loop had paid for itself in its first year of Core. The goods receipt loop would take half as long to reach Bound. And the company now knew how to do this.



Like solar panels on a factory roof: *months of cost before any saving, then each unit costs a third as much, and they pay back within a year.*

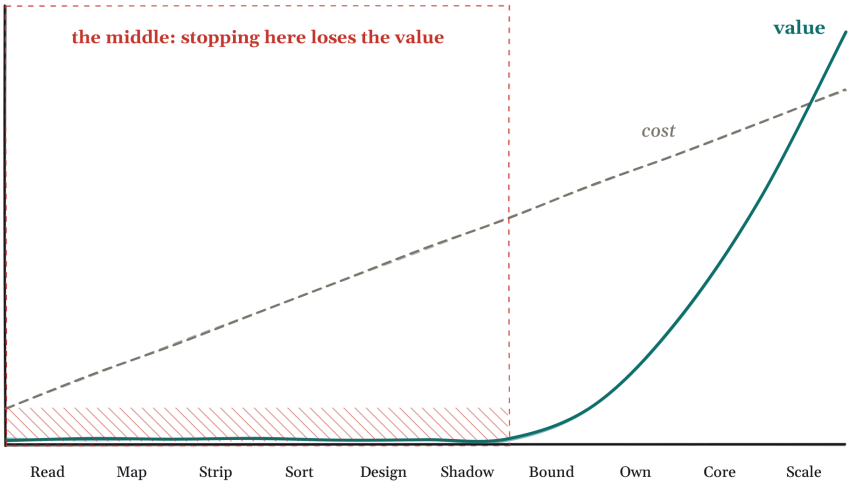


FIGURE 22 What finishing is worth. Cost accumulates from the first step, but value arrives only when the executor acts, at Bound, and grows at Core. A programme that stops before Bound spends most of its cost for almost none of its value.

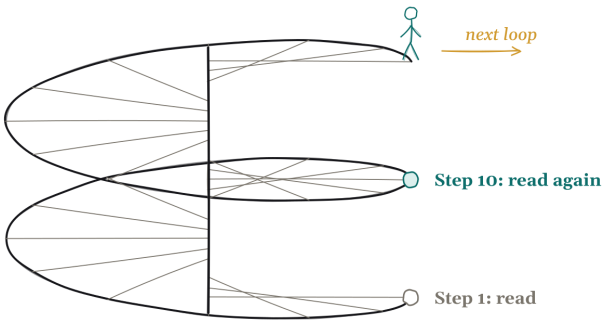
The Cycle Closes

The lifecycle ends where it began. Step 10's map review is the same document that Step 1 produced, one cycle later. Its last act is to choose the next loop to read.

That is not a stylistic choice. It reflects how companies learn. The first reading of a map is a set of guesses: about which loops matter, where their value sits, how ready they are. Each cycle replaces some guesses with evidence. The distribution of a loop's cases, read once at Step 1, is read again at Step 10 with the body handled and the tail exposed. Readiness scores move as neighbouring loops are embedded. The sponsor's view of where to start becomes a view of what the company has learned it can do.

The steps in between follow a pattern that is older than any of the technology involved. Read the real world before abstracting it. Reduce what you see to the few rules and decisions that matter. Name them, so people can find them again. Test them against real cases. Let people build judgement from the results. Then go back to the real world, which will have changed. Business engineering is that loop, practised deliberately. The embedding lifecycle is that loop, applied to the work of putting an executor that decides inside a company's core.

The middle is where most programmes stall today. It will not stay that way. Companies that learn to cross it once will cross it again, faster, and the distance between them and the companies still running pilots will grow with every loop. The difference will not be the models, which everyone can buy. It will be the sequence, followed and signed.



Like a spiral staircase: *each turn comes back above the same spot, one floor higher, and from each landing you choose where the next turn leads.*

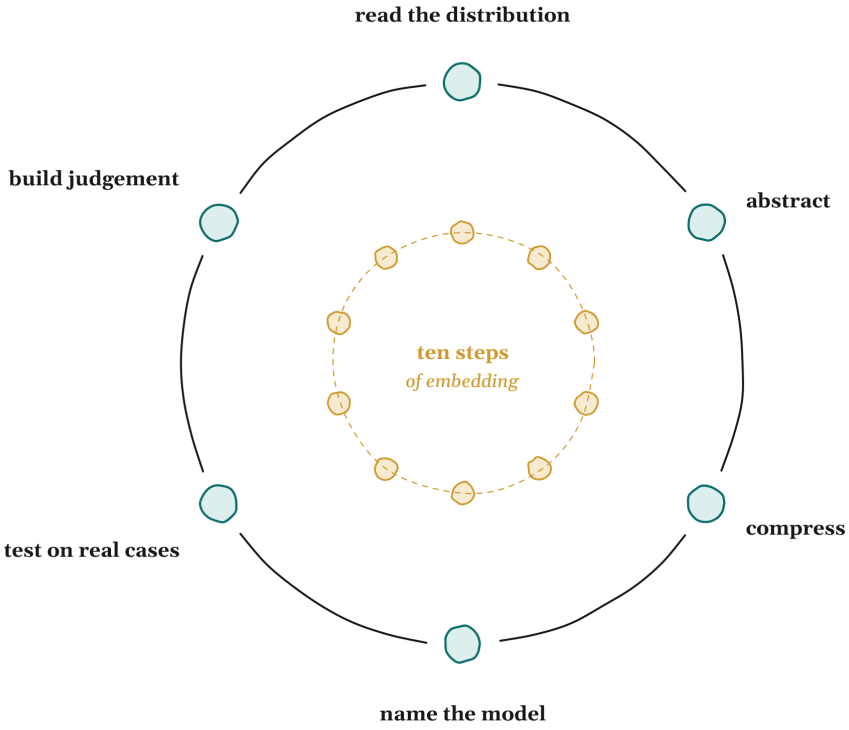


FIGURE 23 The cycle closes. Business engineering read the real distribution, abstracted and compressed it, named the model, tested it against real cases and built judgement. The lifecycle applies that loop to embedding work.

Appendix A: The Lifecycle on One Page

Phase	Step	Question	Signed	Signer	Gate
Read and Map	1. Read	Where does value sit, and which loop first?	Work Loop Map	Executive sponsor	First loop chosen, owner named
Read and Map	2. Map	How does the work run today, at what cost?	Process map, baseline sheet	Process owner	Map recognised, baseline measured
Redesign	3. Strip	Which steps exist only for old constraints?	Strip register	Process owner	Every step struck or kept, constraint named
Redesign	4. Sort	Which rules apply, and who decides what?	Rule register, decision register, contested-decision log	Process owner; leaders for contested decisions	No contested decision open
Redesign	5. Design	What is the loop, with junctions and standard?	Work Loop Canvas, junction register, standard	Loop owner	G0 Mapped, G1 nodes signed
Prove	6. Shadow	Does it match people without acting?	Shadow report	Loop owner and grader	G1 Shadow
Prove	7. Bound	Can it act alone inside written limits?	Gate review record	Control owner	G2 Bounded
Prove	8. Own	Are people named, is the cost measured?	Loop charter, economics sheet	Loop owner; finance	G3 Owned
Run and Scale	9. Core	Does it run in the business, watched?	Operating review, switch-off plan	Loop owner	G4 Core
Run and Scale	10. Scale	What changes across the map, and what next?	Map review, role change plan	Executive sponsor	Next Read begins

Appendix B: Who Signs What

Each role signs at the steps where its authority is needed, and nowhere else. "Consulted" means the role reviews the document before it is signed.

Role	Signs	Consulted
Executive sponsor	Work Loop Map (1), contested decisions (4), map review and role change plan (10)	Economics sheet (8)
Process owner, later loop owner	Process map and baseline (2), strip register (3), rule and decision registers (4), canvas, junction register and standard (5), shadow report (6), loop charter (8), operating reviews and switch-off plan (9)	Every step
Leaders who share a contested decision	The contested-decision log entries they settle (4)	Design (5)
Grader	Shadow report (6)	Standard (5), operating reviews (9)
Control owner	Gate review record (7), each widening of scope (9)	Junction register (5)
Internal audit	None	Gate review (7), operating reviews (9)
Finance (planning and analysis)	Economics sheet (8)	Baseline (2)
Business engineer	None; prepares every document and runs the steps	Every step

Appendix C: Entry and Exit Checklists

Each step starts when its entry conditions are true and ends when its exit conditions are true. A step that cannot meet its exit conditions loops back, or waits.

Step	Enter when	Leave when
1. Read	A sponsor has asked, and the people who know the work have time	Loops listed and placed; candidate distributions read; first loop chosen; owner agreed; map signed
2. Map	Loop chosen, owner agreed	Map drawn with operators and walked through with real cases; workarounds marked; baseline measured on a real sample; both signed
3. Strip	Map and baseline signed	Every step struck or kept; every constraint named; strip ratio stated; register signed
4. Sort	Strip register signed	Rules gathered from documents, systems and people; contradictions decided; every decision classed; every contested decision settled and signed
5. Design	Registers signed, contested log closed	Context with keepers; actions with undo answers; junctions with who, what evidence, how reversed; switch-off holder named; exception queue with owner and time limit; handoffs agreed; standard written; G1 nodes signed
6. Shadow	G0 reached, G1 nodes signed	Several weeks of parallel running; agreement on body and tail separately; every disagreement explained; changes recorded; grader calibrated; report signed
7. Bound	G1 passed	Limits written from evidence; G2 nodes signed; permissions match actions; switch-off tested; weeks of bounded running graded; gate review signed
8. Own	G2 passed	Every role filled by a named person; changes to people's work agreed; learning path for new staff; cost per accepted outcome measured over a cycle, body and tail separately; charter and economics signed
9. Core	G3 passed	Scope widened in recorded bands; drift checks running; operating reviews held and signed; switch-off tested on schedule; all twelve nodes signed
10. Scale	A loop at G3 or G4	

Step	Enter when	Leave when
		Handoffs and exceptions read for the next loop; distribution re-read; map updated; next loops chosen; role change plan signed

Appendix D: The Case in Numbers

The composite manufacturer's supplier invoice loop, from Read to the first map review. All figures are illustrative and set within published ranges.

The distribution read at Step 1

Invoice value	Share of invoices	Share of spend
Under €5,000	81%	14%
€5,000 to €50,000	16%	40%
Over €50,000	3%	46%

Baseline and result

Measure	Baseline (Step 2)	Measured at Own (Step 8)
Invoices per month	7,400	7,400
Cost per accepted invoice, fully loaded	€11.80	€4.10 (body €2.30, tail €14.60)
Days from receipt to payment-ready	16	3.4
Right first time	71%	93%
Invoices raising an exception	24%	9%
Steps in the loop	14	8
Clerks on the loop	9	4

The strip register (Step 3)

Step	Status	Constraint
Print and scan PDF invoices	Struck	Scanning desk predated email intake
Budget owners sign printed batches on Fridays	Struck	Approvals were once on paper
Weekly batch posting	Struck	The ERP posting job once ran weekly
Second clerk re-checks invoices under €1,000	Struck	An error spike years ago, long resolved
Clerks chase goods receipts by email	Struck	

Step	Status	Constraint
		Receiving teams could not see waiting invoices
Manual duplicate check against a spreadsheet	Struck	Duplicate rule now applied by the ERP
Treasury call-back on changed bank details	Kept	Payment fraud risk still applies
Match invoice to order and receipt	Kept	Required by the outcome

The decision register (Step 4)

Class	Count	Examples
Mechanical	8	Duplicate detection; match within tolerance; tax-code validation
Bounded	5	Cost-centre coding of small non-order invoices; partial delivery within limits
Consequential	3	Invoices over €50,000; changed bank details; non-order invoices over €5,000
Contested	1, settled	Paying a disputed invoice to protect a single-source supply: allowed up to €25,000 with the procurement director's signature, logged; above that, the CFO

The junction register (Step 5)

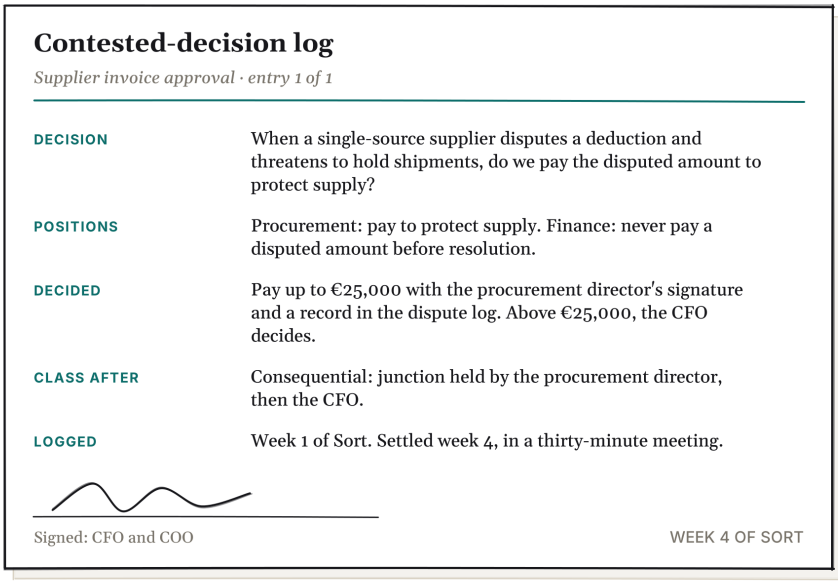
Junction	Who decides	What they see	How it is reversed
Invoices over €50,000	Financial controller	Matched order, receipt, contract price, supplier payment history	Hold before payment run; credit note if paid
New or changed bank details	Treasury analyst	Change request, call-back record on a separately held number	Payment blocked until verified
Non-order invoices over €5,000	Budget owner of the cost centre	Invoice, supplier record, budget position	Hold and return to supplier
Switch-off	Head of accounts payable	Grading dashboard, exception queue	All invoices to the clerks' queue within an hour

Shadow and Bound (Steps 6 and 7)

Measure	Value
Invoices in shadow	9,800 over six weeks
Agreement under €5,000	97.8%
Agreement over €50,000	81%
Rules added after shadow	2
Bounded scope	Order-backed, under €5,000, within 2%, supplier paid for six months or more: 74% of invoices, 12% of spend
Bounded invoices meeting the standard	98.6%
Switch-off test	Cases returned in 20 minutes, none lost

Three signed documents

The figures below show three of the loop's documents as they were signed: the contested-decision log entry from Step 4, the gate review record from Step 7, and the economics sheet from Step 8.



Contested-decision log

Supplier invoice approval · entry 1 of 1

DECISION	When a single-source supplier disputes a deduction and threatens to hold shipments, do we pay the disputed amount to protect supply?
POSITIONS	Procurement: pay to protect supply. Finance: never pay a disputed amount before resolution.
DECIDED	Pay up to €25,000 with the procurement director's signature and a record in the dispute log. Above €25,000, the CFO decides.
CLASS AFTER	Consequential: junction held by the procurement director, then the CFO.
LOGGED	Week 1 of Sort. Settled week 4, in a thirty-minute meeting.

Signed: CFO and COO

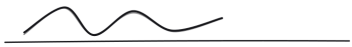
WEEK 4 OF SORT

FIGURE 24 The contested-decision log entry, as signed at Step 4.

Gate review record: G2 Bounded

Supplier invoice approval

SCOPE ALONE	Order-backed invoices under €5,000, within 2% of order price, from suppliers paid for six months or more. 74% of invoices, 12% of spend.
EVIDENCE	Four weeks bounded: 98.6% met the standard. Exception queue cleared within two days every week.
PERMISSIONS	Cut to the actions node. Pilot account able to edit supplier records removed.
SWITCH-OFF	Tested week 1: invoices back with clerks in 20 minutes, none lost.
AUDIT NOTE	Link the dispute log to the payment record before Own.



Signed: Financial controller

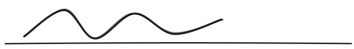
WEEK 4 OF BOUND

FIGURE 25 The gate review record, as signed at Step 7 by the control owner.

Economics sheet

Supplier invoice approval · eight weeks across a quarter-end

BASELINE	€11.80 per invoice, fully loaded, measured at Map on 1,200 invoices.
COST LINES	Platform and model usage; grading and 3% audit; junction time; exception queue; rework; adoption.
RESULT	€4.10 per accepted invoice. Body €2.30. Tail €14.60.
TIME, QUALITY	16 days to 3.4 days, receipt to payment-ready. Right first time 71% to 93%.
ANNUALISED	About €680,000 a year on 7,400 invoices a month, before early-payment discounts.



Signed: Planning and analysis lead

WEEK 8 OF OWN

FIGURE 26 The economics sheet, as signed at Step 8 by finance.

Appendix E: The Lifecycle and Governance Frameworks

Many companies already work within an AI governance framework. The lifecycle does not replace them. It produces the business decisions and signed records those frameworks ask for. This appendix shows where the main ones meet the steps.

The NIST AI Risk Management Framework, published in the United States in January 2023, organises AI risk work into four functions: govern, map, measure and manage. Its "map" function, establishing context and risks, corresponds to Steps 1 to 5. "Measure" corresponds to the shadow evidence, the grading and the economics in Steps 6 to 8. "Manage" corresponds to the junctions, the switch-off and the operating review in Steps 7 and 9. "Govern" runs through the whole lifecycle in its signatures.

ISO/IEC 42001, published in 2023, is a management-system standard for AI. Like other management-system standards, it follows a plan, do, check, act structure and expects documented roles, risk assessment, operational controls, monitoring and improvement. The lifecycle's registers, gate reviews, charters and operating reviews are the kind of documented evidence an audit against the standard looks for.

The European Union's AI Act, adopted in 2024, sets obligations for systems used in high-risk areas, such as decisions about employment, access to credit and essential services. For those systems it requires risk management, logging and effective human oversight by competent people. It also requires those deploying such systems to monitor their operation. A loop that falls in a high-risk area must meet those requirements in law. The junction register, the switch-off plan, the charter naming competent people and the operating review are where the lifecycle produces the evidence. Many loops, such as supplier invoice approval, fall outside the high-risk categories. The same records remain good practice.

None of this is legal advice. Each company should map its loops to the obligations that apply to it, with its own counsel.

The Five Moves

If you do nothing else after reading this book, make these five moves. Each is small, and each removes one of the commonest reasons programmes stall in the middle.

1. Read your map before you choose a loop. Two half-day sessions with the heads of the main functions will list your loops and show which one should go first. Choose by the map, not by the demo.
2. Measure one baseline. Take the loop you most want to change and measure, on a real sample, its cost per case, elapsed time and first-time-right rate. Every later claim depends on it.
3. Hold the contested-decision meeting. Ask the owners of your first loop which decisions two senior people would answer differently. Put those decisions in front of someone with authority, and write down the answers.
4. Run shadow long enough to see the tail. Report agreement for routine cases and consequential cases separately. A single overall figure will mislead you.
5. Put the control owner's name on Bound. The person who signs the executor's limits should be the one whose job is to say no, not the one who wants the loop to run.

The Meters

These are the measures that show whether a lifecycle is healthy. Each is read at the step shown and kept on the operating review afterwards. Measures that count activity, such as cases processed, agents deployed or steps automated, are deliberately left out. They rise whether or not the business is better off.

Meter	What it measures	Read at	Healthy sign
Distribution shape	Share of cases and of value by band	1, 10	Body and tail identified and named
Baseline completeness	Volume, cost, elapsed time, exceptions and first-time-right, all measured	2	Every figure measured on a real sample
Flow efficiency	Hands-on time divided by elapsed time	2, 9	Rises sharply after Strip and Core
Strip ratio	Share of mapped steps struck	3	Every struck step has a named constraint
Contradictions found and decided	Conflicting rules surfaced and settled	4	All decided, none deferred
Contested decisions open	Decisions the organisation never agreed	4 to 9	Zero before Design, and kept at zero
Agreement, body and tail	Shadow results matching people's, by band	6	Both read separately; tail understood
Bounded share	Share of cases, and of value, the executor handles alone	7, 9	Widens in recorded steps
Cost per accepted outcome	Fully loaded cost divided by outcomes meeting the standard	8, 9	Below baseline, body and tail apart
Standard met	Share of outcomes meeting the standard	6 to 9	Stable; drift caught within days
Junction override rate	How often junction holders overrule the executor's evidence	9	Above zero and explained
Exception age	Age of the oldest case in the queue	9	Within the time limit every week
Time per step	Elapsed weeks in each step	All	Falls with each new loop

The Mental Models

Each instrument in this book compresses an idea that recurs across the lifecycle. Here they are with one line each, and how each applies here.

Model	Principle	Applied here
The middle	Most AI value is lost between a working pilot and a process the business relies on	The lifecycle is built for the middle, not for the demo
Inside out	AI creates value inside processes, where companies are least documented	Embedding exposes undocumented rules and decisions, so the steps surface them in order
The executor that decides	Design for an executor that finds its own steps, not one that follows them	The lifecycle designs outcomes, decisions, limits and grading rather than task lists
Body and tail	Most processes have a routine body and a consequential tail	The executor runs the body at Bound; people hold the tail at the junctions
Compensations	Many steps exist for constraints that no longer apply	Strip removes them before anything is designed
Contradictions are findings	Where rules disagree, the process has been guessing	Sort records each contradiction and has an owner decide it
Contested decisions	Some decisions were never agreed, and no design can settle them	The gate after Sort stays shut until every one is decided and signed
The junction	A person decides where an action is costly or cannot be undone	Junctions are placed by reversibility, with evidence and a way back
Two kinds of machinery	Learned models are flexible without guarantees; written rules are precise but never complete	The model is bought; the written half is built in Steps 2 to 5
The limits of rules	No rule system of real power covers every case	The tail, tacit rules and contested decisions go to people, who extend the rules
Isomorphism	A map means something only when it mirrors the work	Maps are drawn from walk-throughs and checked against the log
Requisite variety	Only variety can absorb variety	Rules take the body, the model the middle, people the tail
The good regulator	Controlling a system requires a model of it	The record set is the regulator, not paperwork

Model	Principle	Applied here
Falsifiable gates	A test is worth running only if it can fail	Targets are written before Shadow and Bound begin
Reflexivity	A running loop changes the distribution it reads	Scale re-reads before choosing the next loop
Scale follows signatures	A loop is ready when its parts are signed, not when it feels ready	Each gate opens on named signatures
The signer who can say no	Controls are signed by the person whose job is to refuse	The control owner, not the loop owner, signs Bound
Per accepted outcome	Cost means fully loaded cost divided by results that met the standard	Own measures it against the baseline, body and tail apart
Drift is silent	Quality moves without errors as inputs and models change	Core grades every outcome and reviews monthly
Value arrives late	Embedding saves little until the executor acts	Stopping before Bound spends most of the cost for almost none of the value
The cycle closes	Every lifecycle ends at the map that starts the next	Scale feeds Read; the company learns to embed

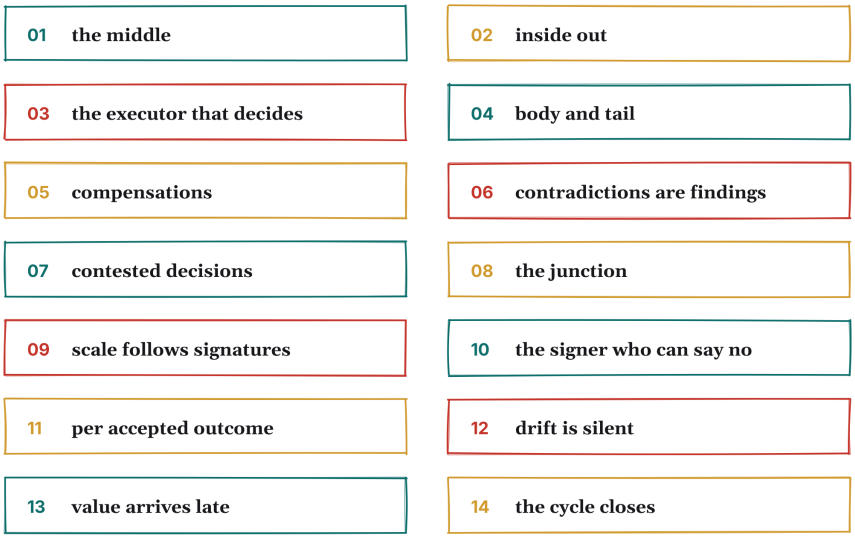


FIGURE 27 The mental models of the lifecycle, read in order from the middle to the cycle that closes.

Glossary

Term	Meaning in this book
Baseline	The measured volume, cost per case, elapsed time, exception rate and first-time-right rate of a loop before it changes
Body	The large share of routine, similar cases in a loop's distribution
Bounded decision	A decision needing judgement inside a written limit; the executor may take it alone
Business engineer	The person who runs the lifecycle for a loop, preparing documents and chasing signatures
Compensation	A step that exists only to work around a constraint
Consequential decision	A decision with a high cost of error or an irreversible action; a person always decides
Contested decision	A decision the organisation has never agreed on; leaders must settle it in writing
Control owner	The person responsible for the company's controls over a process; signs the gate review at Bound
Distribution	How a loop's cases spread by value, difficulty or risk
Drift	A gradual change in the quality of outcomes as inputs or models change
Embedding	An executor that decides running part of a core process, inside written limits, with named people answering for it
Exception queue	The single place where cases outside every rule go, with an owner and a time limit
Flow efficiency	Hands-on time divided by elapsed time
Gate	The point at the end of a step where a named person decides whether the work moves on
Good regulator	Conant and Ashby's result that controlling a system requires a model of it; the record set is that model
Grader	The person who keeps the standard and the grading calibrated
Isomorphism	A match of structure between a map or register and the work it describes; the condition for the map to mean anything
Junction	A point where a person decides instead of the executor, defined by who decides, what they see and how it is reversed
Loop owner	

Term	Meaning in this book
	The person who answers for a loop's outcome, including the parts the executor produces
Mechanical decision	A decision that follows a written rule completely
Neurosymbolic	Combining learned models with written rules, tools and checks; every embedded loop is such a combination
Reflexivity	The way a running loop changes the distribution of cases it was designed to read
Requisite variety	Ashby's law that a regulator needs as much variety as the cases it faces; spread across rules, the model and people
Scale Gates	G0 Mapped, G1 Shadow, G2 Bounded, G3 Owned, G4 Core
Signature	A named person's dated, recorded acceptance of a document
Standard	The written criteria every outcome is graded against, and the method of grading
Strip ratio	The share of a map's steps struck as compensations
Switch-off	The tested means by which a named person stops the loop and returns cases to people
Tail	The small share of cases that carries most of the value or risk
Work Loop Canvas	The one-page description of a loop in twelve connected nodes
Work Loop Map	All of a company's loops placed by value to the core and readiness
Work loop	The unit that produces one outcome end to end

Notes and Sources

The middle. MIT NANDA, *The GenAI Divide: State of AI in Business 2025* (July 2025), based on a review of more than 300 public AI initiatives, 52 interviews and 153 survey responses. It reported that about 5% of custom enterprise AI tools reached production, as summarised in Virtualization Review, 19 August 2025. McKinsey & Company, *The state of AI in 2025: Agents, innovation, and transformation* (November 2025; fieldwork June to July 2025). In it, 88% of respondents reported regular AI use in at least one function, and about one third reported scaling AI. Some 23% reported scaling an agentic system somewhere in the enterprise, and no more than 10% in any single function. Gartner, press release, 25 June 2025: "Over 40% of agentic AI projects will be canceled by the end of 2027," due to escalating costs, unclear business value or inadequate risk controls. Gartner, July 2024: at least 30% of generative AI projects expected to be abandoned after proof of concept by the end of 2025.

Foundations. George Boole, *An Investigation of the Laws of Thought* (1854). Kurt Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I," *Monatshefte für Mathematik und Physik* 38 (1931). Michael Polanyi, *The Tacit Dimension* (1966). Douglas R. Hofstadter, *Gödel, Escher, Bach: An Eternal Golden Braid* (1979), on isomorphism, levels and strange loops. Henry Kautz, "The Third AI Summer," AAAI Robert S. Engelmore Memorial Lecture (2020), and Artur d'Avila Garcez and Luís C. Lamb, "Neurosymbolic AI: The 3rd Wave," *Artificial Intelligence Review* (2023), on combining learned and symbolic methods. Trieu H. Trinh et al., "Solving olympiad geometry without human demonstrations," *Nature* 625 (2024), on AlphaGeometry.

Control and measurement. Norbert Wiener, *Cybernetics* (1948). W. Ross Ashby, *An Introduction to Cybernetics* (1956), on requisite variety. Roger C. Conant and W. Ross Ashby, "Every Good Regulator of a System Must Be a Model of That System," *International Journal of Systems Science* 1, no. 2 (1970). Karl Popper, *The Logic of Scientific Discovery* (1959). George Soros, *The Alchemy of Finance* (1987), on reflexivity. G. K. Chesterton, *The Thing* (1929), on the fence. Charles Goodhart, "Problems of Monetary Management: The U.K. Experience" (1975). Lisanne Bainbridge, "Ironies of Automation," *Automatica* 19, no. 6 (1983). Fabrizio Dell'Acqua et al., "Navigating the Jagged Technological Frontier," Harvard Business School Working Paper 24-013

(2023). *The Business Engineering Body of Knowledge* develops each of these foundations at length.

Plan-do-check-act. Walter A. Shewhart, *Statistical Method from the Viewpoint of Quality Control* (1939). W. Edwards Deming's lectures in Japan from 1950 popularised the cycle.

The waterfall. Winston W. Royce, "Managing the Development of Large Software Systems," *Proceedings of IEEE WESCON* (1970). Royce recommended iteration and building a preliminary version, advice later readers largely ignored.

Stage-gate. Robert G. Cooper, "Stage-Gate Systems: A New Tool for Managing New Products," *Business Horizons* 33, no. 3 (1990).

CRISP-DM. Pete Chapman et al., *CRISP-DM 1.0: Step-by-step Data Mining Guide* (2000), from a consortium formed in 1996 that included DaimlerChrysler, SPSS and NCR.

Service management and MLOps. The IT Infrastructure Library was developed by the UK Central Computer and Telecommunications Agency from the late 1980s. D. Sculley et al., "Hidden Technical Debt in Machine Learning Systems," *Advances in Neural Information Processing Systems* 28 (2015).

The stop cord. Taiichi Ohno, *Toyota Production System: Beyond Large-Scale Production* (1988, English edition), on jidoka and stopping the line.

Invoice baselines. Ardent Partners, *Accounts Payable Metrics that Matter in 2025*, as reported by Thomson Reuters. It gives an average cost to process an invoice of about $9.40 and an average cycle time of 17.4 days, against about 3.1 days for the best. Exception rates were 22% for most organisations against 9% for the best. APQC Open Standards Benchmarking: a median of 75% of invoice line items matched the first time, and about 12,000 invoices processed per accounts payable employee per year. The composite manufacturer's figures are illustrative and set within these ranges.

Spend concentration and payment fraud. Tail spend is commonly reported as about a fifth of spend across most suppliers (The Hackett Group). Association of Certified Fraud Examiners, *Occupational Fraud 2024: A Report to the Nations*: billing schemes appeared in 22% of cases, with a median loss of $100,000. Association for Financial Professionals, *2025 Payments Fraud and Control Survey*: 79% of organisations faced attempted or actual payments fraud in 2024.

Customer-facing loops. In *Moffatt v. Air Canada*, 2024 BCCRT 149, a Canadian tribunal held the airline responsible for incorrect refund information given by the chatbot on its website.

Governance frameworks. National Institute of Standards and Technology, *Artificial Intelligence Risk Management Framework (AI RMF 1.0)* (January 2023). ISO/IEC 42001:2023, *Information technology: Artificial intelligence: Management system*. Regulation (EU) 2024/1689 (the AI Act), in particular Articles 9 (risk management), 14 (human oversight) and 26 (obligations of deployers).

The Thread

This book threads into the rest of The Business Engineer Library through a handful of shared instruments. Each is defined here in its own terms, and each is developed further elsewhere.

The Work Loop Canvas and the Work Loop Map are the instruments every step fills. *The Work Loop Canvas* describes each of the twelve nodes and the century of work design behind them.

Strip and sort are the core of the redesign. *The Redesign* takes them further, with eight passes over a documented process, the controls that survive and invert, and what a process map cannot tell you.

The standard is how every outcome is graded. *The Standard* treats acceptance as a commercial question: what enterprise AI has to prove, and who decides. *Evaluation Engineering* covers the grading machinery in technical depth.

The junction and the charter are how people stay answerable. *The Agentic Organization* covers how roles, teams and accountability change when many loops run at once.

The drift check is how quality is watched in the core. The engineering volumes describe what makes a system drift and how it is monitored beneath the business.

Also in This Series

This book is part of a series on embedding AI into the enterprise core. Each volume stands alone, and together they form one discipline.

The Business Engineering Body of Knowledge. Volume I, the foundation: from the laws of thought and neurosymbolic AI to closed loops of action, the core method of business engineering, and every concept and measure of the discipline defined once.

The Work Loop Canvas. Volume II, the instrument: one loop on one page, its twelve nodes, and the century of designing work that led to it.

The Embedding Lifecycle. Volume III, this book: the ten steps from the first reading of a company's loops to an executor running in the core.

The Work Loop Library. Volume IV, the record: a library of work loops across six chains, six of them taken through the full lifecycle with every template filled in.

The New Business Engineer. The four volumes brought together into one book on the discipline of embedding AI into the enterprise.

THE BUSINESS ENGINEER

The Embedding Lifecycle

Read before you build. Prove before you act. Sign every gate.

businessengineer.ai