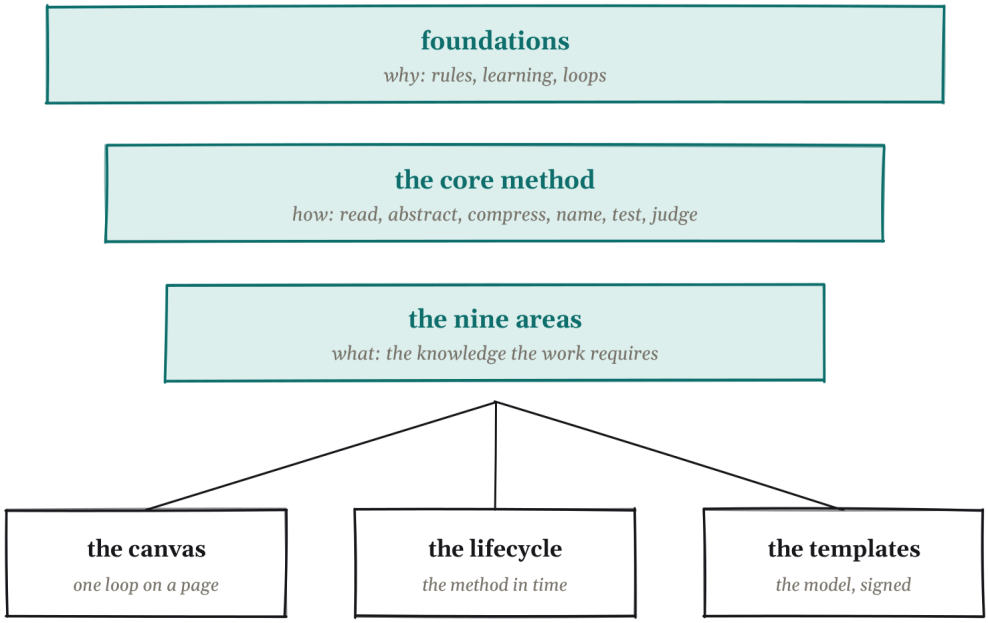
The Business Engineering Body of Knowledge

From the Laws of Thought to Closed Loops of Action



> Read the distribution. Write the rules.
> Close the loop.

Gennaro Cuofano

The Business Engineering Body of Knowledge

From the Laws of Thought to Closed Loops of Action: The Foundations, Concepts and Measures for Embedding AI in the Enterprise

The Business Engineering Body of Knowledge
From the Laws of Thought to Closed Loops of Action: The Foundations, Concepts and Measures for Embedding AI in the Enterprise

The Business Engineer Library · The Embedding Series, Volume I

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

What a Discipline Needs 7

Part One: Foundations **11**

I. The Laws of Thought 12

II. The Limits of Rules 15

III. Strange Loops and Levels 19

IV. The Neural Turn 23

V. Neurosymbolic AI 26

VI. Closed Loops 30

VII. Why This Matters for the Enterprise 34

Part Two: The Core **37**

VIII. What Business Engineering Is 38

IX. Reading the Distribution 42

X. How Distributions Are Made 45

XI. Abstraction 48

XII. Compression 51

XIII. Mental Models 54

XIV. Testing and Judgement 57

XV. The Closed Loop of Business Engineering 60

Part Three: Reading the Business **64**

XVI. The Shape of This Body of Knowledge 65

XVII. Distributions 69

XVIII. Value and the Map 76

XIX. Work and Process 84

Part Four: Designing the Loop **92**

XX. Rules and Decisions 93

XXI. The Executor 101

XXII. Control and Acceptance 109

Part Five: Owning the Result 118

XXIII. People and Accountability 119

XXIV. Economics 127

XXV. Governance 134

Part Six: Across the Areas 140

XXVI. Notation 141

XXVII. The Measures 146

XXVIII. The Competencies 149

XXIX. The Evidence Base, as Read in 2026 154

XXX. How the Discipline Cascades 157

Part Seven: The Mental Models Library 161

XXXI. How the Models Library Works 162

XXXII. Models for Every Loop 165

XXXIII. Heavy-Tailed Loops 175

XXXIV. High-Volume Routine Loops 183

XXXV. Loops That Decide About People 191

XXXVI. Regulated and Audited Loops 199

XXXVII. Customer-Facing Loops 207

XXXVIII. Loops That Depend on Outside Parties 214

XXXIX. Loops Whose Steps Are Generated 221

XL. Loops in a Chain 229

Appendix A: Areas Against Steps 236

Appendix B: Instruments by Step 237

Appendix C: Reading Map 238

Appendix D: Foundational Reading 239

Appendix E: One Loop Through Every Area 240

Appendix F: The Thirty-Nine Loops by Type 241

The Five Moves 244

The Mental Models 245

Glossary 248

Notes and Sources 253

The Thread 261

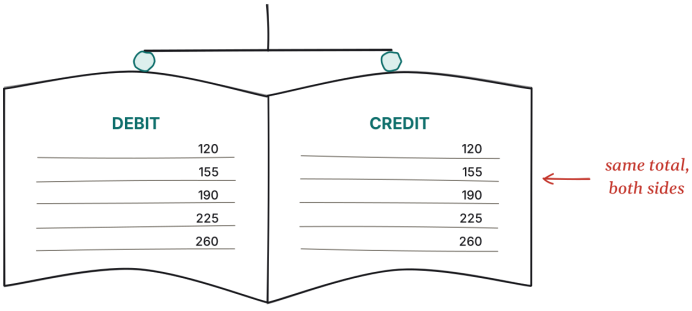
Also in This Series 262

What a Discipline Needs

In 1494, a Franciscan friar named Luca Pacioli published a mathematics textbook in Venice. Twenty-seven of its pages described a method Venetian merchants had used for generations: recording every transaction twice, once as a debit and once as a credit, so that the books always balanced. Pacioli did not invent double-entry bookkeeping. He wrote it down, named its parts, and explained them so that someone who had never sat in a Venetian counting house could learn them.

That is roughly how a practice becomes a discipline. People who do the work develop methods that succeed. For a long time those methods live in their heads and their habits, passed on by apprenticeship. Then someone writes down the concepts, gives them stable names, defines how they are measured, and shows how they fit together. After that, the practice can be taught, compared, audited and improved by people who never met its originators.

Embedding AI into the core of a company is at that stage now. A small number of companies have learned to move an AI system that makes decisions from a pilot into work the business relies on. Their methods are real, and they converge. But they live mostly in practitioners' heads, in slide decks and in vendors' marketing. The same idea goes by five names in five companies, and measures that sound alike are calculated in incompatible ways.



Like a merchant's ledger: *once every entry was written twice and its parts were named, anyone could check the books, not only the merchant who kept them.*

Why the foundations come first

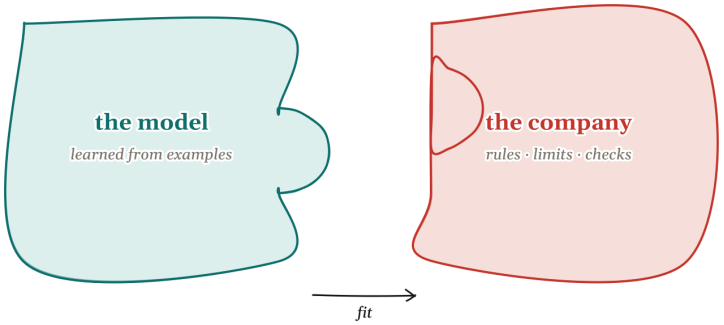
There is a second reason this body of knowledge matters, and it is deeper than tidiness. The machines now entering companies are the latest step in a very

old project: writing down how thinking works, so that something other than a person can do it.

That project has had two traditions. One tried to capture thought in explicit rules and symbols, from the logic of Aristotle to the algebra of George Boole and the expert systems of the 1980s. The other tried to let machines learn from examples, from the first artificial neurons of the 1940s to the language models of today. Each tradition succeeded at something the other could not do, and each failed where the other was strong. The most capable systems now combine them. Researchers call the combination neurosymbolic AI.

A company that embeds AI into its core is building exactly such a combination, whether it knows it or not. The model brings the learned, pattern-matching half. The company has to bring the other half: its rules, its decisions, its limits, its standards and the loops that check results against the world. That half is what this book describes. **A body of knowledge for embedding AI is, in a precise sense, the symbolic layer of the company.**

So the book begins with the foundations. It explains what rules can and cannot do, what learning from data can and cannot do, and why the combination works. It also explains how closed loops connect any of it to real action. Then it sets out the core concepts of business engineering, the method by which a practitioner reads a business and models it well enough to act on it. Only then does it turn to the nine areas of knowledge the work requires. Everything else in the discipline, from the canvas to the lifecycle to the templates, follows from these pages.



Like two pieces of a jigsaw: *the model brings the learned half, and the company has to cut the other half, its rules, limits and checks, to fit.*

How the book is organised

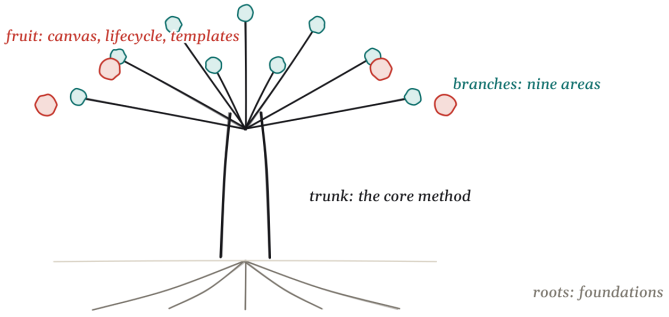
Part One, Foundations, traces the two traditions from the laws of thought to neurosymbolic AI, and adds the science of closed loops. It ends with what all of this means for an enterprise.

Part Two, The Core, sets out the method of business engineering. It covers reading distributions, how distributions are made, abstraction, compression, mental models, testing and judgement, and the closed loop that joins them.

Parts Three to Five hold the nine knowledge areas. Each covers one kind of knowledge the work requires, opens with how it connects to the foundations, places its concepts on the embedding lifecycle, and then defines them one entry at a time.

Part Six covers what runs across the areas: the notation, the exact definitions of the measures, the competencies of practitioners, the state of the evidence, and how the rest of the discipline cascades from this book.

Part Seven is a library of mental models, organised by the type of loop in which each matters most. It covers every loop, heavy-tailed loops, high-volume routine loops and loops that decide about people. It also covers regulated and audited loops, customer-facing loops, loops that depend on outside parties, loops whose steps are generated, and loops in a chain.

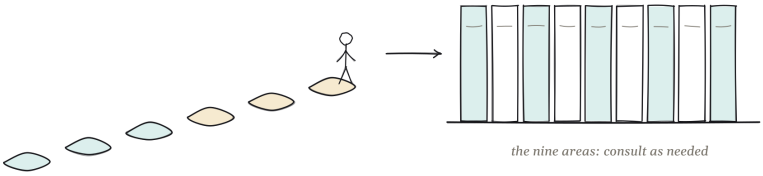


Like a tree: *the foundations are the roots, the core method is the trunk, the nine areas are the branches, and the working tools are the fruit.*

How to read it

Read Parts One and Two in order. They are an argument, and each chapter builds on the one before. After that, the book is designed to be consulted. A single composite company, a mid-size European maker of industrial pumps and valves, provides the examples, so the same loops and figures recur across

areas. Its figures are illustrative, set within published ranges. The book assumes no technical background, and no knowledge of any other book.



Parts One and Two: in order

the nine areas: consult as needed

Like learning a new city: *walk the first two parts in order once, then use the rest as a map you return to whenever you need a street.*

PART ONE

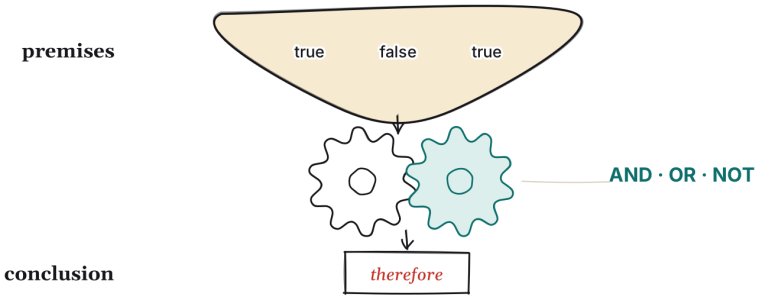
Foundations

I

The Laws of Thought

In 1854, a self-taught mathematician in Cork named George Boole published a book with an ambitious title: *An Investigation of the Laws of Thought*. His aim was to show that reasoning itself could be written as algebra. Statements could be treated as variables that were either true or false, and combined with operations such as "and", "or" and "not". If the rules of combination were right, correct conclusions would follow mechanically from correct premises, with no judgement required.

Boole was continuing a very old line of work. Aristotle had catalogued the valid forms of argument more than two thousand years earlier. In the seventeenth century, Gottfried Leibniz imagined a universal language of concepts and a "calculus of reasoning". With them, he thought, disputes could be settled by calculation instead of argument: "let us calculate". Boole made a large part of that dream concrete.

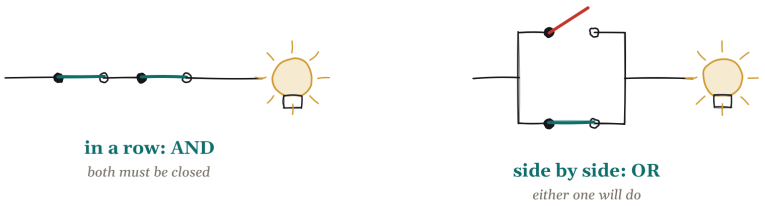


Like a mill: *pour in true premises, turn the handle of the rules, and correct conclusions come out, with no judgement needed along the way.*

From algebra to machines

Boole's algebra waited eighty years for its most consequential use. In 1937, a graduate student at the Massachusetts Institute of Technology named Claude Shannon noticed that electrical relays behaved like Boole's variables. A relay was open or closed, and relays wired in series or in parallel computed "and" and "or". His master's thesis, *A Symbolic Analysis of Relay and Switching Circuits*, showed that any logical expression could be built as a circuit. Every digital computer since rests on that observation. Boole's laws of thought became the laws of the machine.

For the next several decades, the dominant idea of intelligence in computing followed directly. To make a machine think, write down the symbols and the rules, and let it calculate. In 1956, a workshop at Dartmouth College launched the field of artificial intelligence on largely this premise. Its proposal stated that every feature of intelligence could in principle be described so precisely that a machine could simulate it.

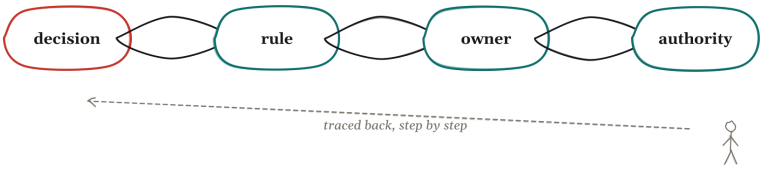


Like a hallway light with two switches: *wire them one after the other and the bulb needs both; wire them side by side and either will do.*

What the symbolic tradition got right

The symbolic tradition gave us things no business could now do without. Rules written explicitly can be read, checked, audited and changed. A rule can say exactly why a decision was taken. Two people reading the same rule should reach the same answer. Logic guarantees that if the premises hold, the conclusion holds. Accounting systems, tax engines, eligibility checks, approval matrices and every workflow configured in enterprise software are this tradition at work.

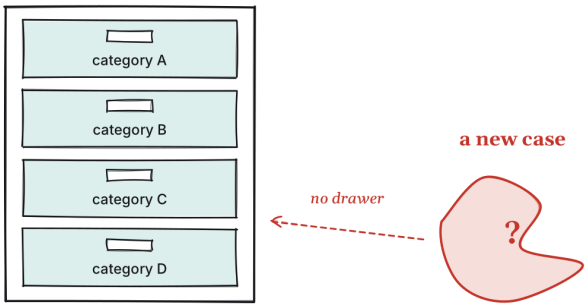
Its idea of reliability is the one auditors, regulators and boards still use. A decision is trustworthy if it can be traced to a rule, the rule to an owner, and the owner to an authority. **The registers, standards and gates of this book are the symbolic tradition applied to a company's own work.**



Like a chain of custody: *every decision links to a rule, every rule to an owner, every owner to an authority, and anyone can follow the chain back.*

What it assumed

The symbolic tradition made a quiet assumption: that the world could be described completely in advance. Every case would fit a category, every category would have a rule, and every rule would be consistent with the others. In mathematics, that assumption was about to be tested to destruction. In business, it had never been true at all.



Like a filing cabinet built in advance: *it works perfectly until a case arrives that fits no drawer.*

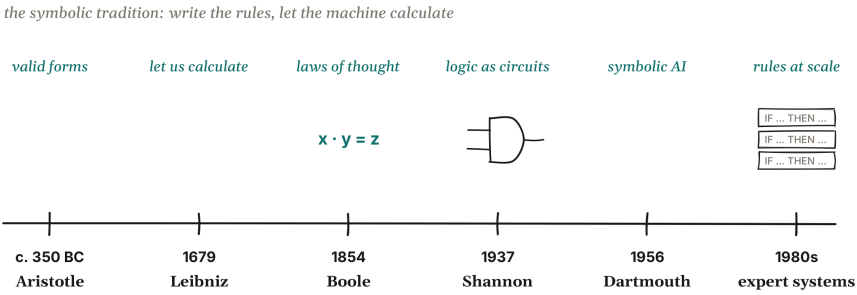


FIGURE 1 The laws of thought. From Aristotle's syllogisms to Boole's algebra to Shannon's relay circuits: reasoning written as rules, then built into machines.

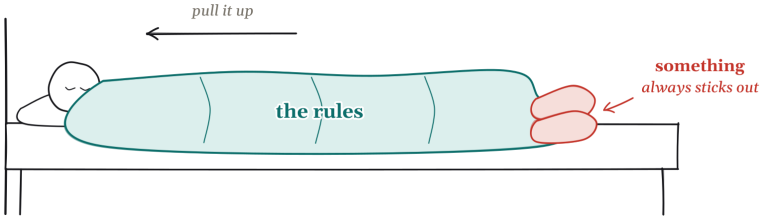
II

The Limits of Rules

At the start of the twentieth century, the mathematician David Hilbert set out a programme to put all of mathematics on a complete and consistent set of rules. Every true statement would be provable from the axioms, and no contradiction could ever be derived. It was the symbolic dream at its most ambitious.

In 1931, a twenty-five-year-old logician named Kurt Gödel showed that it could not be done. Any consistent formal system powerful enough to express ordinary arithmetic contains true statements that the system cannot prove. Adding the missing statement as a new rule does not help: the larger system has its own unprovable truths. Five years later, Alan Turing showed a related limit for machines. There is no general procedure that can decide, for every program, whether it will ever finish running.

Gödel's result is about mathematics, and it should not be stretched into a slogan about everything. But it carries a lesson that applies by analogy to any attempt to capture a complex domain in rules. A rule system is a map. However detailed it becomes, it can be complete only for a world simpler than the one it describes.



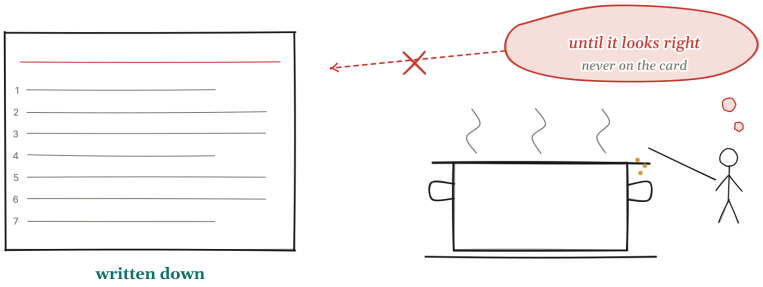
Like a blanket that is too short: *pull it up to cover the shoulders and the feet stick out, so every rulebook leaves some truth uncovered.*

The same limit, in business

Companies met a practical version of this limit long before they met AI. In the 1970s and 1980s, expert systems tried to capture specialist knowledge as thousands of if-then rules. MYCIN diagnosed blood infections at Stanford. XCON configured computer orders at Digital Equipment Corporation, and for a time saved it millions of dollars a year. These systems worked well inside

their domains and failed abruptly at the edges. They also became very expensive to maintain as rules multiplied and began to conflict.

Edward Feigenbaum, one of the field's founders, named the central difficulty the knowledge acquisition bottleneck. Experts could not simply state what they knew. Much of their knowledge was tacit. The philosopher Michael Polanyi had put it in a single sentence in 1966: "we can know more than we can tell".



Like a family recipe card: *it lists every step, but the cook's sense of when it looks right never made it onto the card.*

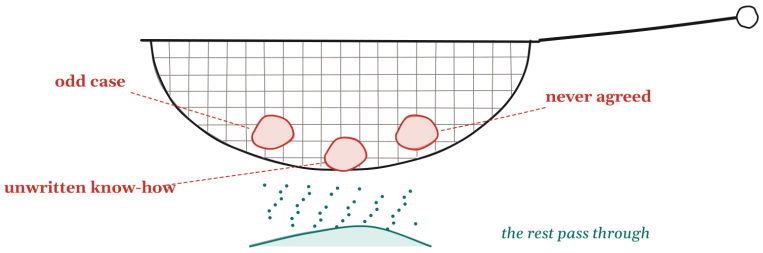
Three limits a business engineer meets

Every embedding programme meets three forms of the same limit.

The first is the tail. However many rules a loop has, some cases fall outside all of them: the new supplier with an unusual tax treatment, the claim that fits two categories, the customer whose history changes the answer. Rules handle the body of a distribution well and the tail badly, because the tail is where the world is richer than the map.

The second is tacit knowledge. Many of the rules a loop really follows live in experienced people's heads, and they cannot all be written down. Some can be drawn out by asking, by walking real cases and by watching what people do. Others show up only as judgement.

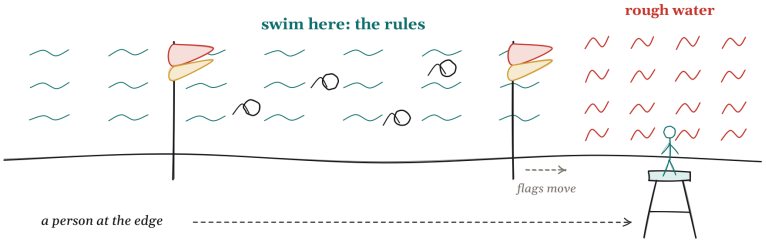
The third is the contested decision. Some questions cannot be settled inside the rule system at all, because the organisation never agreed on the answer. No amount of further rule-writing resolves them. Someone has to step outside the system and decide. In Gödel's terms, the system has to be extended from outside, and only a person with authority can do that.



Like a kitchen sieve: *most cases pass straight through the rules, but three kinds stay behind, the odd case, the unwritten know-how and the question nobody settled.*

What follows for design

The limits of rules are not an argument against rules. They are an argument for knowing where rules stop. A well-designed loop writes down every rule it can and sorts every decision. It also says plainly what happens at the edges: which cases go to people, who decides what was never agreed, and how the rules are extended when the world shows them a new case. Much of this book is about drawing that boundary well.



Like a beach with flags: *swim freely between them, a lifeguard is posted where the safe water ends, and the flags move when the sea changes.*

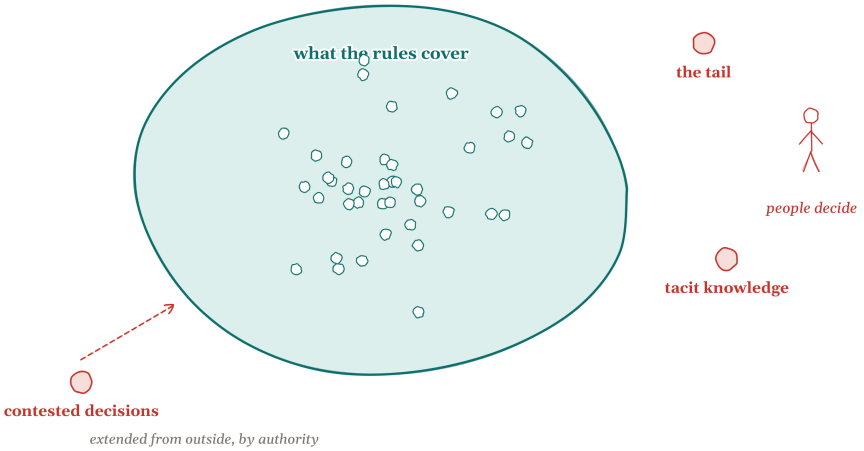
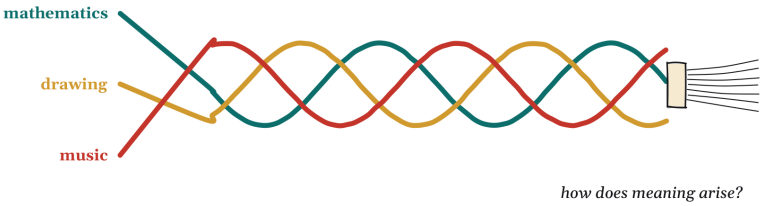


FIGURE 2 The limits of rules. A rule system covers the body of cases well; the tail, tacit knowledge and contested decisions lie outside it, where people must decide and the rules must be extended from outside.

III

Strange Loops and Levels

In 1979, the cognitive scientist Douglas Hofstadter published *Gödel, Escher, Bach: An Eternal Golden Braid*. It won the Pulitzer Prize and became one of the most influential books ever written about minds and machines. It braids together the mathematics of Gödel, the drawings of M. C. Escher and the music of Johann Sebastian Bach. Through them it asks how meaning and selves can arise from systems of meaningless symbols. Four of its ideas are directly useful to anyone designing work for machines.

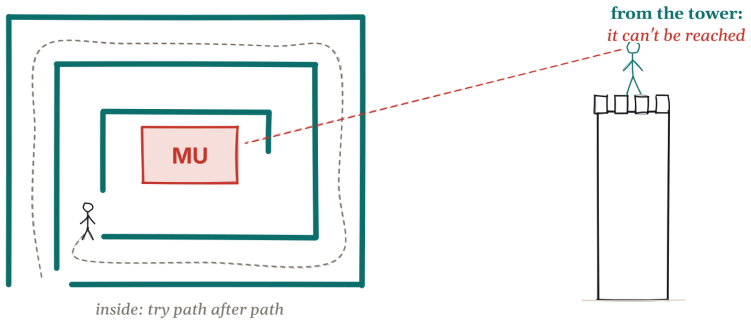


Like a three-strand braid: *mathematics, drawing and music woven into one rope that asks how meaning can grow out of symbols that mean nothing alone.*

Formal systems and meaning

Hofstadter opens with a simple puzzle. Begin with the string MI. Apply a few mechanical rules: add a U after a final I, double everything after the M, replace III with U, drop UU. Can you produce MU? Working inside the rules, people try string after string. The answer comes only from stepping outside the rules and noticing a property of them: the number of Is is never divisible by three, so MU is unreachable.

The lesson is about the difference between working inside a system and reasoning about it. A rule-following executor works inside. Knowing whether the rules can ever produce the result you want requires a view from outside. In a company, that outside view belongs to the people who design and own the loop.

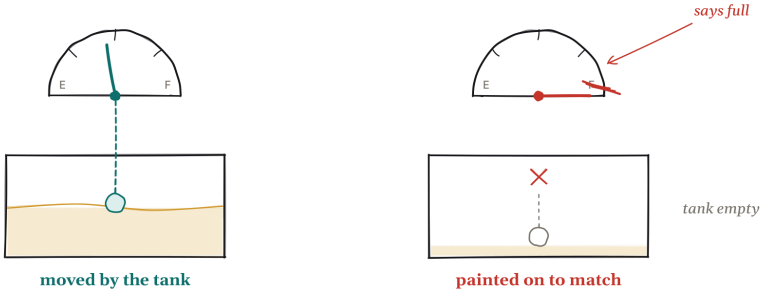


Like a hedge maze: *walkers inside try path after path, while someone on the tower sees at once that the centre can never be reached.*

Isomorphism: how symbols come to mean something

A string of symbols means nothing by itself. It acquires meaning when its structure mirrors the structure of something else, so that operations on the symbols track changes in the world. Hofstadter calls this isomorphism. Numbers mean quantities because arithmetic mirrors counting. A ledger means a company's finances because its entries mirror real transactions.

For embedding, this is the central question. A rule register, a decision register or a standard means something only if it mirrors how the work actually runs. A register copied from the procedure manual mirrors the work as it was designed, not as it is. That is why the discipline insists on walking real cases and measuring real baselines. The symbols must be anchored in the world, or they are only a formal game.

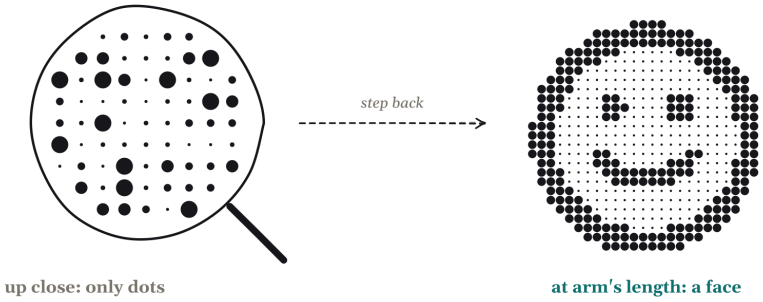


Like a fuel gauge: *the needle means something only because a float in the tank moves it; a needle painted on to match the manual tells you nothing.*

Levels

Hofstadter shows that the same system can be described at many levels, and that each level has its own useful laws. An ant colony behaves intelligently although no single ant does. A brain thinks although no single neuron thinks. Explanation at the wrong level is not wrong, but it is useless.

Businesses are the same. A case can be described as keystrokes, as process steps, as decisions, as a loop, or as a line in the financial model. The skill of abstraction, which Part Two describes, is largely the skill of choosing the level at which a problem becomes tractable.

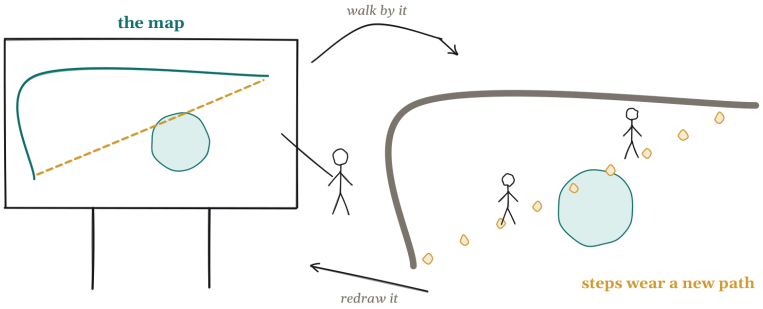


Like a newspaper photo: *under a magnifier it is only dots, at arm's length a face; the skill is picking the distance at which the problem makes sense.*

Strange loops

The book's central image is the strange loop. Moving through the levels of a system, you find yourself back where you started. A system that represents itself becomes able to change itself. Hofstadter argued that this kind of self-reference is the root of what we call a self. Escher's drawings of hands drawing each other and staircases that climb forever are visual versions of it.

A company that embeds AI well develops a strange loop of its own. It builds a model of its work: the map, the registers, the standard. It acts through that model, and the action changes the work. The changed work then changes the model, through the exceptions, the drift and the map review. The organisation comes to represent itself, and the representation becomes part of how it runs. **The method of business engineering is the discipline of keeping that loop honest:** anchored in real cases, revised when the world changes, and owned by people who answer for it.



Like a park map on a signboard: *walkers follow it, their steps wear a new shortcut, and the keeper redraws the map to match the ground.*

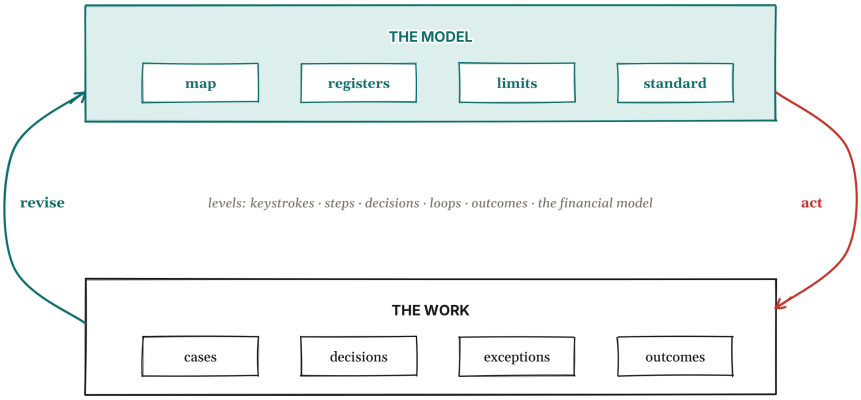


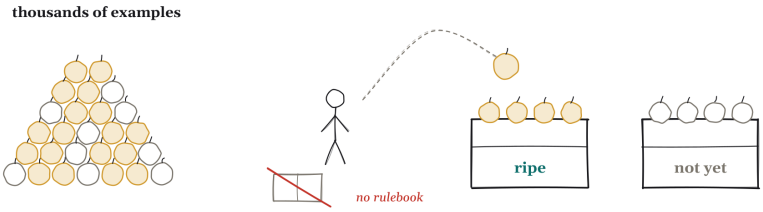
FIGURE 3 Strange loops and levels. The company builds a model of its work, acts through it, and the action changes the work; the changed work changes the model. Kept honest, the loop is how the company learns.

IV

The Neural Turn

Alongside the symbolic tradition ran another one, almost as old. In 1943, the neurophysiologist Warren McCulloch and the logician Walter Pitts proposed a simplified model of a neuron. It took inputs, weighed them and fired if the total passed a threshold. They showed that networks of such units could compute logical functions. In 1958, Frank Rosenblatt built the perceptron, a machine that learned to classify images by adjusting its weights from examples rather than following written rules.

The idea was radical. Instead of telling the machine the rules, show it examples and let it find the regularities itself. It went through long winters. In 1969, Marvin Minsky and Seymour Papert showed the limits of simple perceptrons, and interest collapsed for more than a decade. It revived in 1986, when David Rumelhart, Geoffrey Hinton and Ronald Williams popularised a method for training networks with many layers. It revived again, decisively, after 2012, when deep networks trained on large datasets with fast hardware began to outperform every other approach to images, speech and text.

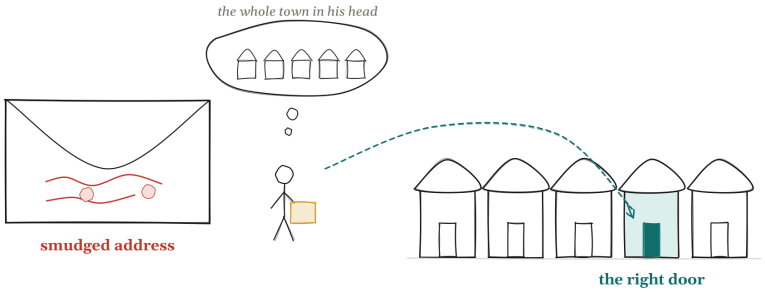


Like an apprentice at a fruit stall: *nobody hands her a rulebook; she watches thousands of apples sorted until she can tell a ripe one herself.*

Learning a distribution

A neural network does not store rules. It stores a compressed representation of the patterns in the data it was trained on. A language model, trained to predict the next word across a large share of the world's text, ends up encoding a great deal about grammar, facts, styles of reasoning and how situations usually unfold. Researchers have shown formally that good prediction and good compression are two sides of the same thing. In a precise sense, a language model is a compression of a distribution.

That explains both its strengths and its weaknesses. Its strength is that it handles the messy, unstructured, never-quite-the-same inputs that defeated rule systems. It can read an invoice in a layout it has never seen, understand an email written in anger, and summarise a contract nobody templated. It covers ground the symbolic tradition could not.

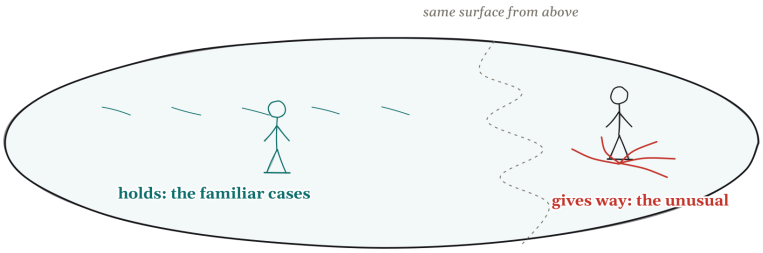


Like a seasoned postman: *years of rounds have packed the whole town into his head, so even a smudged, misspelled address still reaches the right door.*

What it does not give you

Its weaknesses mirror the strengths of rules. A neural model offers no guarantee that its output follows any particular rule. It can be confidently wrong, especially on cases unlike those it learned from. Its competence is uneven in ways that are hard to predict. One study of consultants called this a "jagged frontier": quality rose sharply on tasks inside it and fell on tasks just outside, where people trusted answers they should have checked. And a model cannot, by itself, say why it decided as it did in terms an auditor would accept.

Put simply, a neural model is excellent at the body of a distribution it has seen and unreliable at the tail it has not. That is the opposite trade to rules. Rules are precise but brittle at the edges, and learned models are flexible but unaccountable. **Neither is enough on its own for work a business must answer for.**



Like a frozen lake: *the ice holds firm over most of it, but its edge is jagged and the thin patches look exactly like the thick ones.*

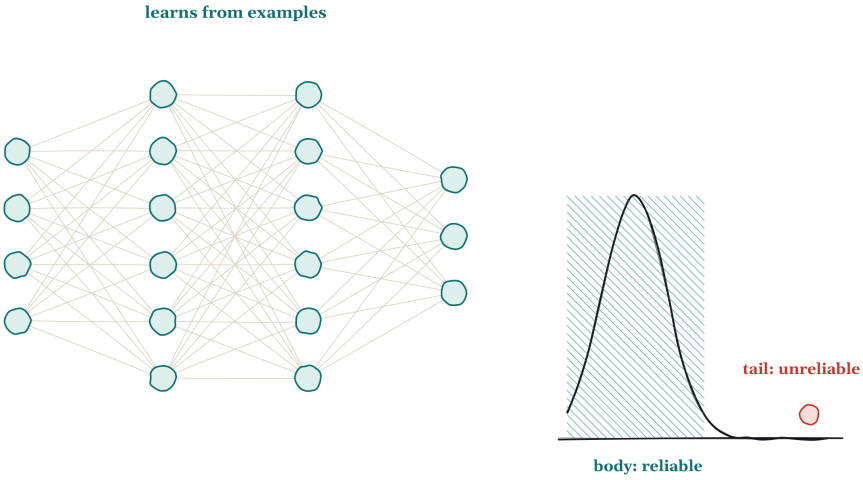
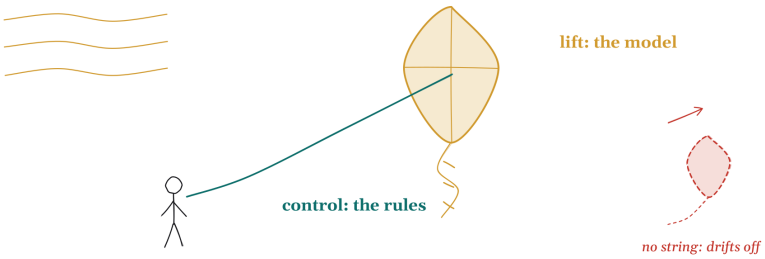


FIGURE 4 The neural turn. Instead of writing rules, show the machine examples and let it compress the patterns. A learned model is strong on the body of the distribution it saw and unreliable on the tail it did not.

V

Neurosymbolic AI

The obvious response to two traditions with complementary strengths is to combine them. Researchers have argued for this for decades, and in the 2020s it became the leading direction for systems that must be both capable and reliable. The approach is called neurosymbolic AI.

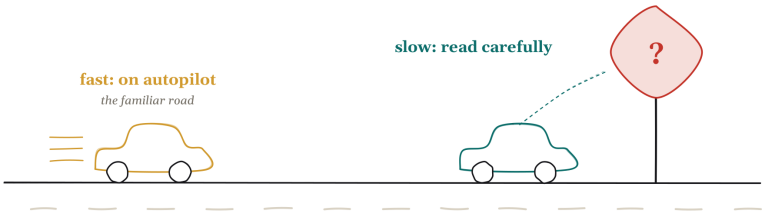


Like a kite and its string: *the kite gives lift from the wind, the string gives control, and on its own either one just falls or drifts away.*

Two systems

A helpful analogy comes from psychology. In *Thinking, Fast and Slow*, published in 2011, Daniel Kahneman described two modes of human thought. System 1 is fast, automatic and associative: recognising a face, reading a mood, sensing that a number looks wrong. System 2 is slow, deliberate and rule-following: checking a calculation, applying a policy, constructing an argument. People rely on System 1 for most of what they do and call on System 2 when the stakes or the doubts are high.

The analogy is loose, but useful. A learned model behaves somewhat like System 1: fast pattern recognition over rich, unstructured input. Symbolic components behave like System 2: explicit rules, checks, logic and records. A capable system needs both, and the design question is how they hand work to each other.



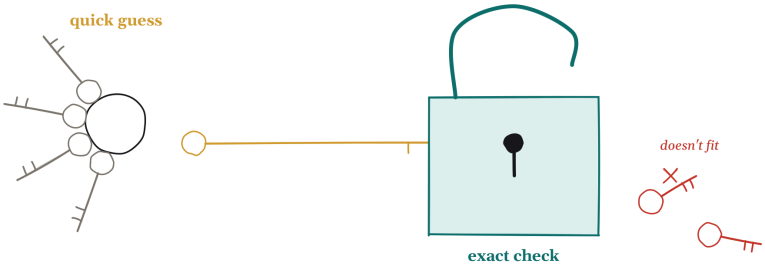
Like driving a familiar road: *hands steer on autopilot until an odd sign appears, then the driver slows down and reads it carefully.*

Ways of combining them

Researchers have catalogued several ways to join the two halves. Henry Kautz, in a widely cited 2020 lecture, set out a taxonomy that ranges from symbolic systems calling neural components to neural systems calling symbolic tools. Artur d'Avila Garcez and Luís Lamb described neurosymbolic AI as the field's "third wave", after symbolic AI and deep learning. Gary Marcus argued that robust AI would need explicit knowledge and reasoning alongside learning.

There are now striking demonstrations. In 2024, DeepMind's AlphaGeometry solved olympiad geometry problems at close to gold-medal level. A language model proposed promising constructions, and a symbolic engine checked and completed the proofs. Neither half could do it alone.

In enterprise systems, the most common combination is simpler and already everywhere. A language model reads and interprets. It then calls tools that apply rules exactly, write to systems of record and check results. A grader verifies the output against a written standard. People decide where the stakes demand it.



Like a locksmith with a ring of keys: *a quick hand guesses which key might fit, and the lock itself tests each guess exactly.*

The enterprise loop as a neurosymbolic system

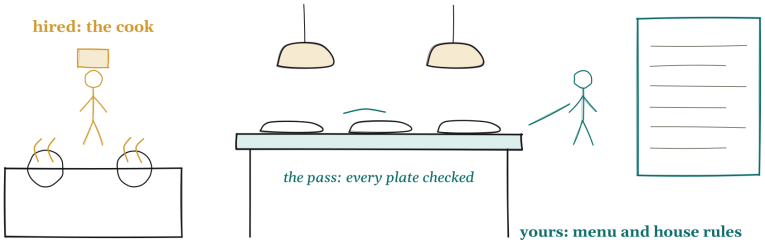
Seen this way, the parts of an embedded loop fall into place.

The neural half is the model: it reads context, interprets cases, proposes decisions and drafts outputs. It covers the variety of the real world that rules cannot.

The symbolic half is what the business writes down. It includes the rule register and decision register, which say what must be followed and who may decide what. It includes the bounded limits, which say where the model may act alone. The standard and its rule checks say what counts as a correct result. The junctions and contested-decision log say where people decide, and what was decided when the rules ran out.

The connection between them is the platform and its logs: routing, tools, permissions, the switch-off and the record of every case.

This is why a business cannot outsource the symbolic half to a vendor. A model can be bought. A company's own rules, decisions, limits and standards cannot, because they are facts about the company. Writing them down is the work of this discipline. A company that embeds a model without them has built half a neurosymbolic system and is relying on the half that offers no guarantees.



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

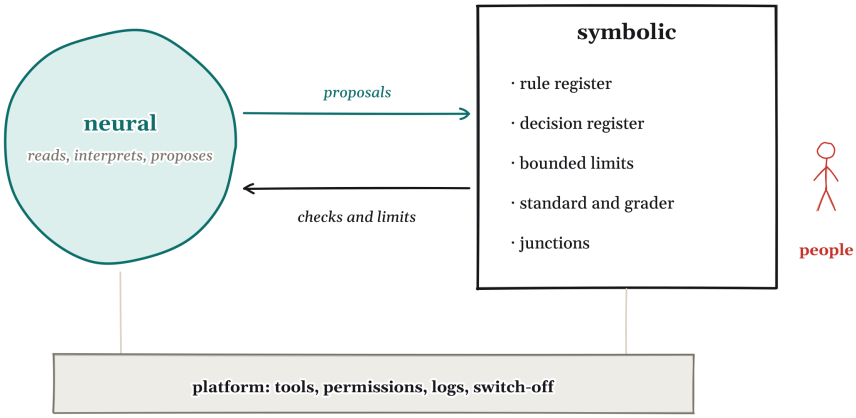
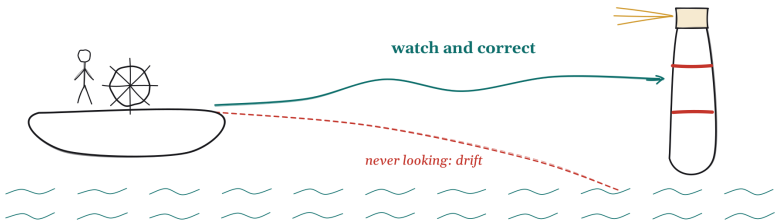


FIGURE 5 Neurosymbolic AI in the enterprise. The model reads and proposes; the symbolic layer of rules, limits, standard and junctions checks, constrains and records; people decide at the edges. The platform connects them.

VI

Closed Loops

Rules and learned models are ways of deciding. Neither, by itself, connects a decision to the world. That connection is the subject of a third body of thought: cybernetics, the science of control and communication in systems that act and sense the results of their action.

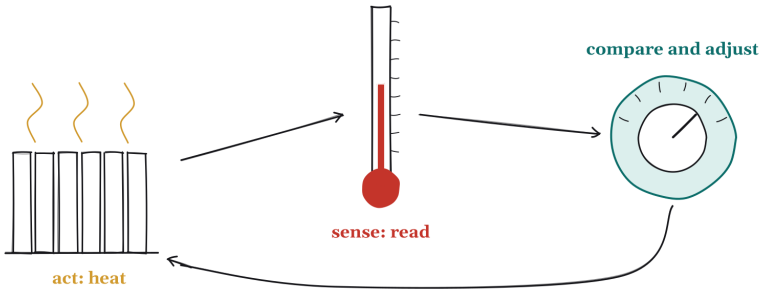


Like a helmsman: *choosing a heading is not enough; he keeps watching the bow against the lighthouse and correcting the wheel as the boat drifts.*

Feedback

In 1948, the mathematician Norbert Wiener published *Cybernetics*. Its central idea was feedback. A system acts, senses the effect of its action, compares it with a goal, and adjusts. A thermostat does this, and so does a pilot. So does any organisation that measures results and changes course. Without feedback, a system can only hope its actions had the intended effect. With it, the system can correct itself.

An embedded loop is a feedback system. The executor acts on cases, and the grader senses whether outcomes meet the standard. The operating review compares results with targets, and people adjust rules, limits and context. Remove the grader or the review, and the loop is running open: acting without knowing whether its actions are right.

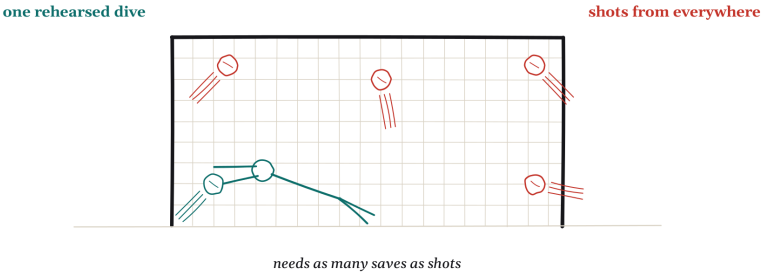


Like a thermostat: *it heats, reads the room, compares the reading with the setting and adjusts; take away the thermometer and the heater runs blind.*

Requisite variety

In 1956, the cybernetician W. Ross Ashby stated what he called the law of requisite variety. A controller can only control a system if it can produce at least as many distinct responses as the disturbances it must handle. Put plainly, only variety can absorb variety.

This law explains much of the design of embedded work. A fixed rule set has limited variety, so it handles the body of a distribution and fails on the tail. A learned model adds a great deal of variety, which is why it handles messy inputs. People, at junctions and in the exception queue, add the variety of judgement for cases nobody anticipated. A loop that tries to handle a heavy-tailed distribution with rules alone violates Ashby's law, and so does a loop that removes the people from its tail.



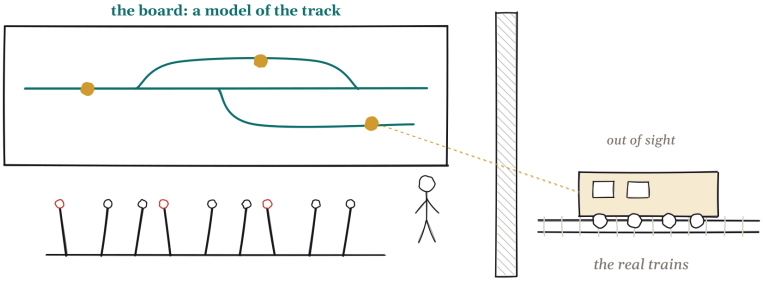
Like a goalkeeper facing shots from every angle: *one rehearsed dive covers one corner, and only as many kinds of save as kinds of shot keeps the goal safe.*

The good regulator

In 1970, Roger Conant and Ashby proved a theorem whose title states its result: "Every good regulator of a system must be a model of that system". To

control something well, you must represent it. A controller that does not model the system it controls can only succeed by luck.

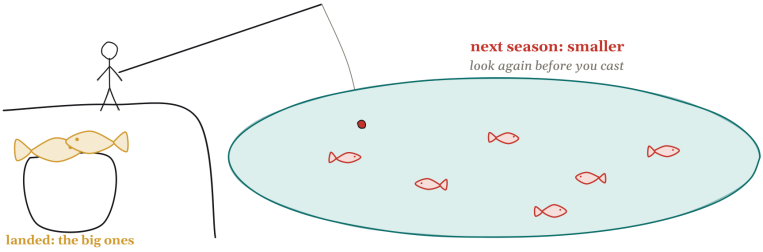
This is the strongest argument for the discipline in this book. A company that wants to control an embedded loop must hold a model of that loop. That means its distribution, its rules, its decisions, its limits, its costs and the way it fails. The map, the canvas, the registers and the standard are that model. **They are not documentation produced for its own sake. They are the regulator.**



Like a railway signal box: *the signaller never sees the trains, only a board that mirrors the track, so without a faithful board safe control is luck.*

Loops that change the world they read

One more property matters. Closed loops in business change the distribution they read. A collections loop that chases invoices faster changes how customers pay. A screening loop that contacts candidates sooner changes who applies. The investor George Soros called this reflexivity in markets: participants′ views change the reality those views are about. Embedded loops are reflexive in the same way. This is why the method never finishes. Read the distribution, act, read it again, because the acting changed it.



Like fishing a lake: *keep landing the big fish and next season the catch is smaller, because the fishing changed the lake you are reading.*

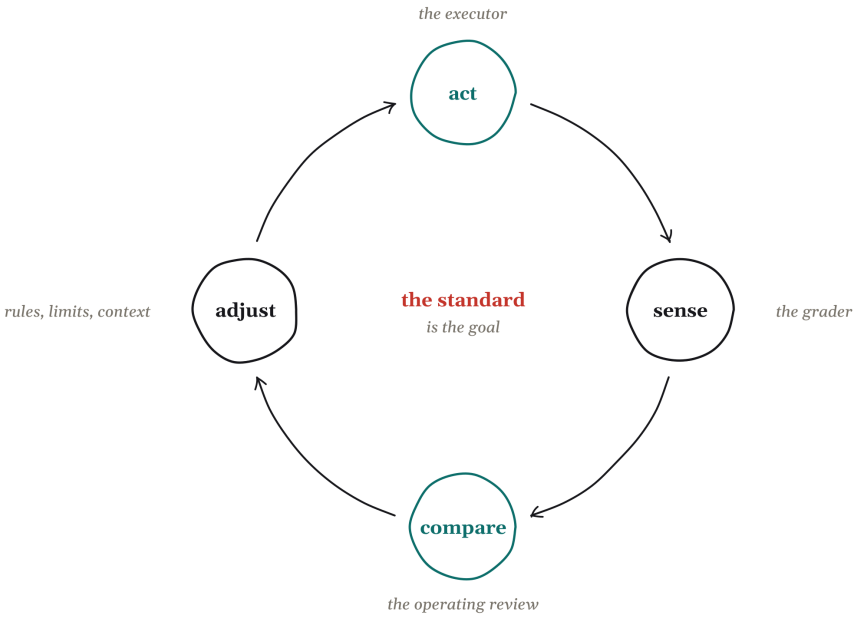
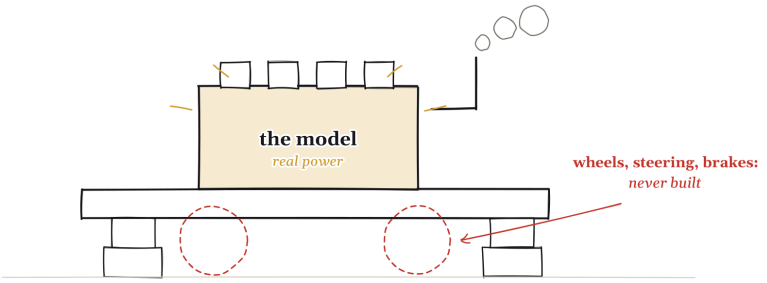


FIGURE 6 Closed loops. Act, sense the result, compare with the goal, adjust. In an embedded loop the executor acts, the grader senses, the operating review compares, and people adjust the rules, limits and context.

VII

Why This Matters for the Enterprise

The previous chapters can now be drawn together into a single argument. It explains why AI stalls in so many companies, and what has to be built for it not to.



Like a powerful engine bolted onto a cart with no wheels: *the power is real, but nothing was built for it to move, so it stalls.*

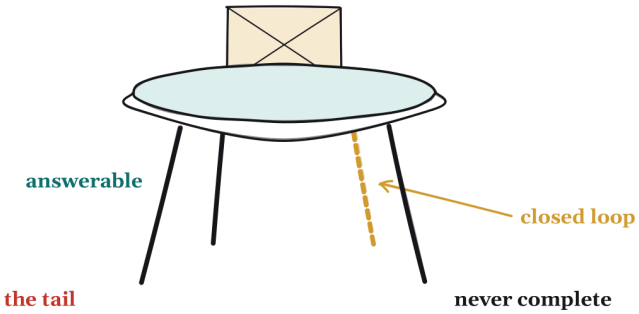
Four claims

First, an executor that decides is a learned model, and learned models are strong on the body of a distribution and unreliable on its tail. They come with no guarantees and no reasons an auditor would accept.

Second, a business's work must be answerable. Decisions must trace to rules, rules to owners, owners to authority. That requires a symbolic layer: rules, decisions, limits and standards written down and signed.

Third, no rule system is complete. Some cases fall outside every rule, some knowledge stays tacit, and some decisions were never agreed. Those edges need people, placed deliberately, and a way to extend the rules when the world shows them something new.

Fourth, control requires a model and a closed loop. To run an embedded loop well, a company must hold a model of it, grade every outcome, feed the results back, and revise the model as the loop changes the world it reads.

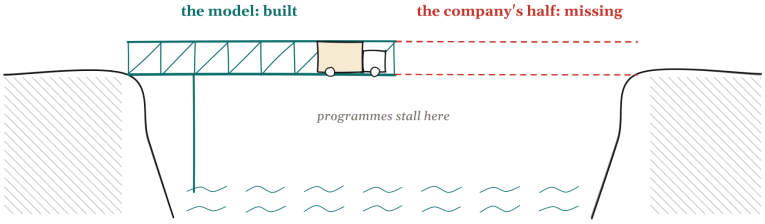


Like a four-legged stool: *the tail, answerable work, incomplete rules and a closed loop each hold a corner, and the seat tips if any leg is missing.*

The conclusion

Together, these claims say that embedding AI into a company's core means building a neurosymbolic, closed-loop system around each unit of work. The model provides the neural half. The company must provide everything else: the symbolic layer, the human junctions at the edges, and the feedback that keeps the whole thing anchored in reality.

Most companies have the first half and little of the rest. Their rules are scattered and contradictory, and their decisions are unsorted. Their edges are undesigned, and their loops run open. That is the gap in which programmes stall. It is not a technology gap. It is a gap in the company's knowledge of itself, written down well enough to regulate an executor that decides.



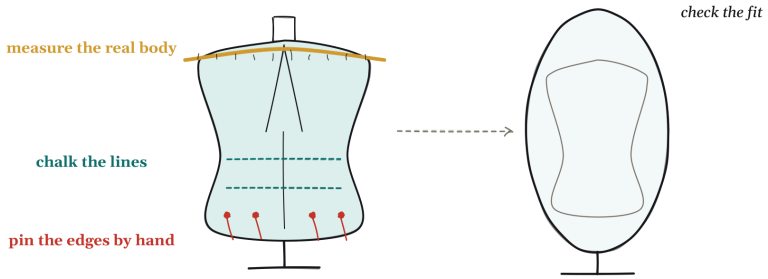
Like a bridge built from one bank only: *the model's half reaches out over the river, and the company's half, where programmes stall, was never built.*

What the business engineer builds

The business engineer is the person who builds the missing half. They read the real distribution of a loop's cases to know where the body ends and the tail begins. They abstract and compress the work into rules, decisions and limits, which become the symbolic layer. They place people where rules run out. They

write the standard and build the feedback that grades every outcome. And they keep the strange loop honest, revising the company's model of itself as the work changes.

The rest of this book describes how. Part Two sets out the core method. Parts Three to Five define the nine areas of knowledge in which the method is applied. Part Six shows how the rest of the discipline cascades from them.



Like a tailor fitting a suit bought off the rack: *measure the real body, chalk the lines, pin the edges by hand and keep checking the fit in the mirror.*

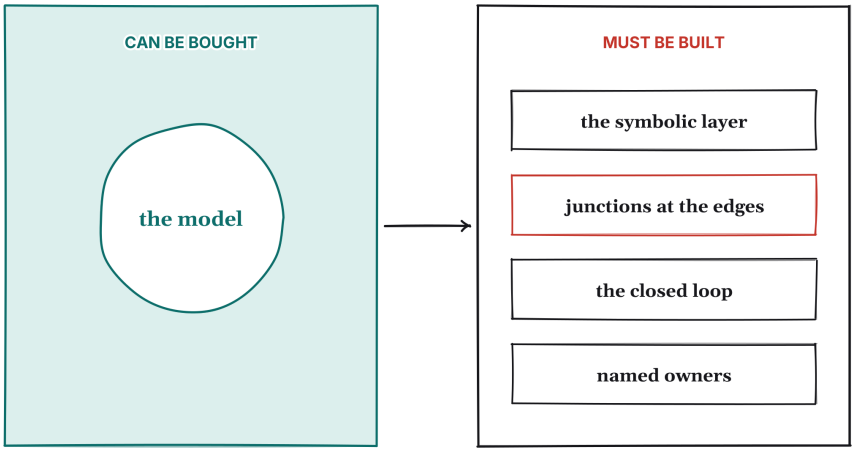


FIGURE 7 Why this matters. The model is the half a company can buy. The symbolic layer, the human junctions at the edges and the closed loop are the half it must build, because they are facts about the company.

PART TWO

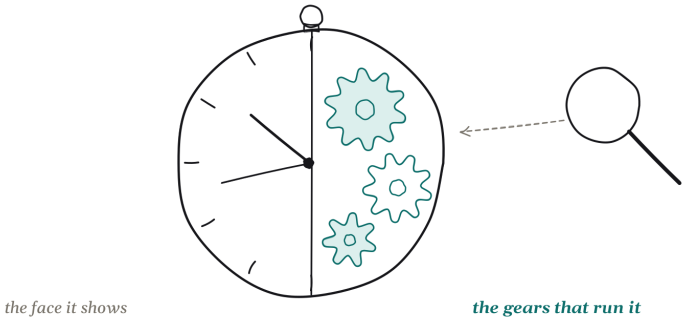
The Core

VIII

What Business Engineering Is

Business engineering is the practice of understanding how a business actually works, well enough to redesign it. It treats a company as a system: a set of loops that turn inputs into outcomes, governed by rules, decisions, people and money. It reads that system from evidence rather than from organisation charts, and it changes it on purpose.

The word "engineering" is chosen deliberately. Engineers work from the properties of the materials in front of them, not from how they wish those materials behaved. They measure before they design, test before they rely on anything, and write down why a design is the way it is. Business engineering applies that temperament to companies. The materials are cases, rules, decisions, people, systems and cash flows. The designs are the loops that produce the company's outcomes.



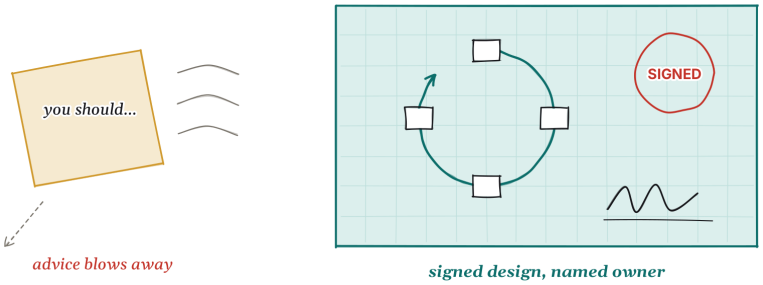
Like a watchmaker who opens the case: *the face only shows the time, so you read how the gears really turn before redesigning anything.*

What it is not

It is not strategy, though it informs strategy. Strategy decides where a company competes and how it wins. Business engineering works one level down, on how the work runs, and it often finds that strategy rests on untrue assumptions about the work.

It is not software engineering, though it works closely with it. Software engineering builds systems. Business engineering decides what those systems should be allowed to do inside the business, and how anyone would know they did it well.

It is not consulting in its traditional form, though consultants can practise it. Its outputs are not recommendations. They are signed designs, measured baselines and loops running in the core, owned by named people inside the company.

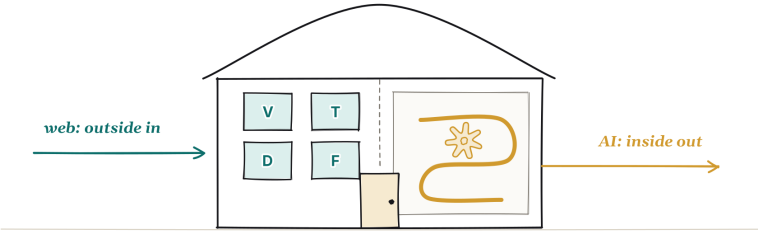


Like a signed blueprint rather than a note of advice: *the advice blows away, while the signed design stays and a named person owns it.*

Outside in and inside out

Business engineering reads a company from two directions. From the outside, it reads the business model through four connected models: value, technology, distribution and financials, abbreviated VTDF. It asks what value the business creates and for whom, and what technology makes that value possible. It asks how the value reaches customers, how the money flows, and how the four reinforce or undermine each other. That view grew up in the platform era, when technology and distribution became the model itself, and outcomes followed power laws.

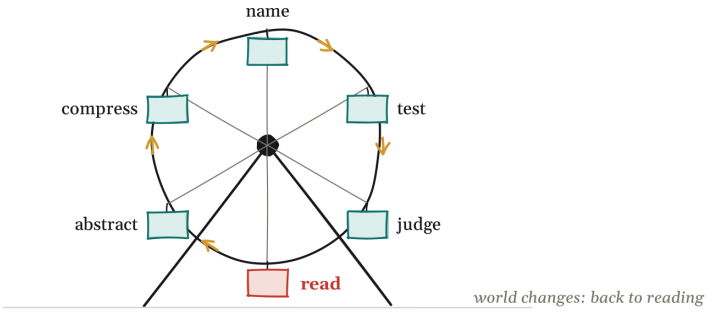
From the inside, it reads the work: the loops, their rules, decisions and costs. The web changed companies from the outside in, through how they reached customers. AI changes them from the inside out, through the work itself. The embedding era makes the inside view as important as the outside one. The two connect wherever a loop's outcome serves a value proposition, its costs roll into the financial model, and its handoffs meet customers.



Like a house: *the web remodelled the front from the street, AI rewires the plumbing from inside, and both meet at the front door.*

The method

At the centre of the discipline is a loop of six moves. Read the real distribution of cases. Abstract the few kinds of case behind the many. Compress the mechanism into the fewest words that still predict. Name it, so it becomes a mental model people can use. Test it against the real world. Build judgement from the results, and return to the distribution when the world changes. The chapters that follow take the moves one at a time, and the last chapter of this part shows how they close into a loop that acts on the world.



Like a Ferris wheel: *six cars come round in turn, read, abstract, compress, name, test, judge, and the wheel returns to reading when the world changes.*

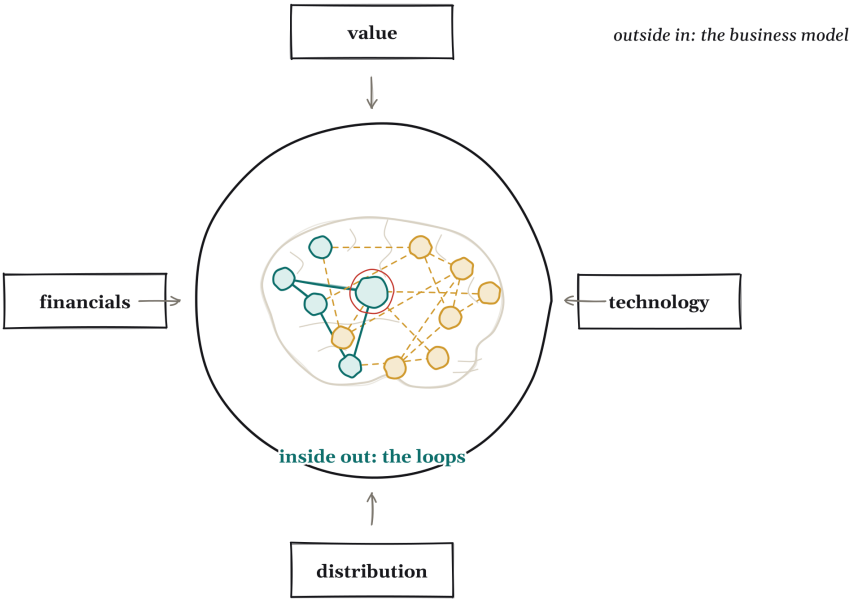
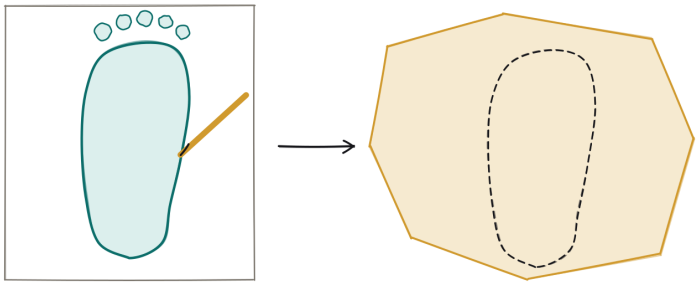


FIGURE 8 Business engineering reads a company from two directions: from the outside, through value, technology, distribution and financial models; from the inside, loop by loop.

IX

Reading the Distribution

The method begins with the real world, before any abstraction. The first question about any piece of work is how its cases are spread. That shape decides more of the design than any other single fact.



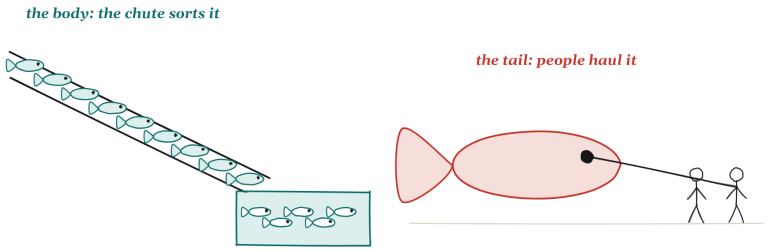
Like a shoemaker who traces the foot before cutting any leather: *the shape of the cases is taken first, and it decides the design.*

Two worlds

Some work produces cases that cluster around a typical value. The heights of adults, the hours on a timesheet and the time to process a routine address change behave like this. Statisticians call the shape normal. In these domains averages mean something, a sample tells you about the whole, and rules work well.

Other work produces cases where a few are enormous and most are small. City sizes, book sales, customer revenue, insurance losses and invoice values behave like this. The usual name for the shape is a power law, and the general term is heavy-tailed. In these domains averages mislead, a sample can miss the cases that matter most, and the largest cases can be larger than all the others put together. The essayist Nassim Taleb called the two worlds Mediocristan and Extremistan.

Most business processes are a mixture: a routine body of similar cases and a thin, heavy tail. That split is the most useful fact in embedding, because a learned model and a set of rules can usually handle the body, while people must hold the tail.

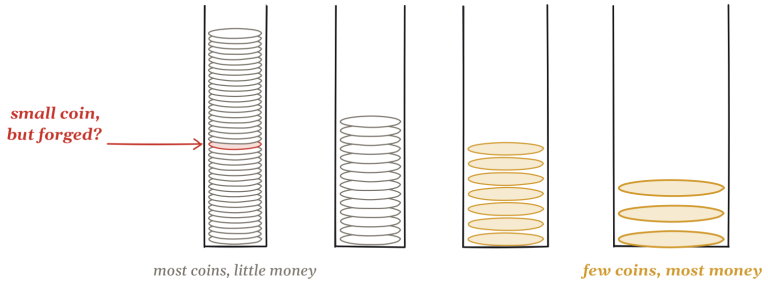


Like a fishing catch: *a sorting chute handles the many small fish of the body, while the rare giant of the tail needs people to haul it in.*

How to read one

Reading a distribution is simple arithmetic and rarely done. Pull a few months of cases. Sort them by the dimension that matters, usually value, sometimes difficulty or risk. Group them into bands and compare, for each band, its share of cases with its share of value. Where the two shares are similar, the work is close to normal. Where a small band carries a large share of value, the tail is heavy.

Then ask the second question: is the distribution the same for value, for difficulty and for risk? Often it is not. A large invoice from a long-standing supplier is high in value and low in difficulty. A small first invoice from a new supplier with new bank details is low in value and high in risk. A good reading names the body and the tail on each dimension that matters.

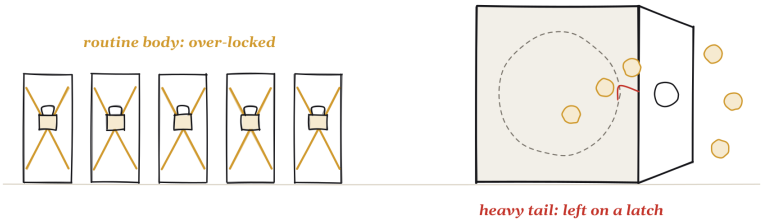


Like sorting a cash box: *compare each tube's share of coins with its share of the money, then check separately for the small forged coin.*

Why it comes first

Every later choice depends on the reading. It decides which loop to start with, what the executor may do alone, where junctions go, how long to test, and how to report results. A design made without it treats every case the same. That

design either over-controls the body and wastes money, or under-controls the tail and loses it.



Like locking a house with one standard lock: *the broom cupboards get chains they never needed, while the vault is left on a flimsy latch.*

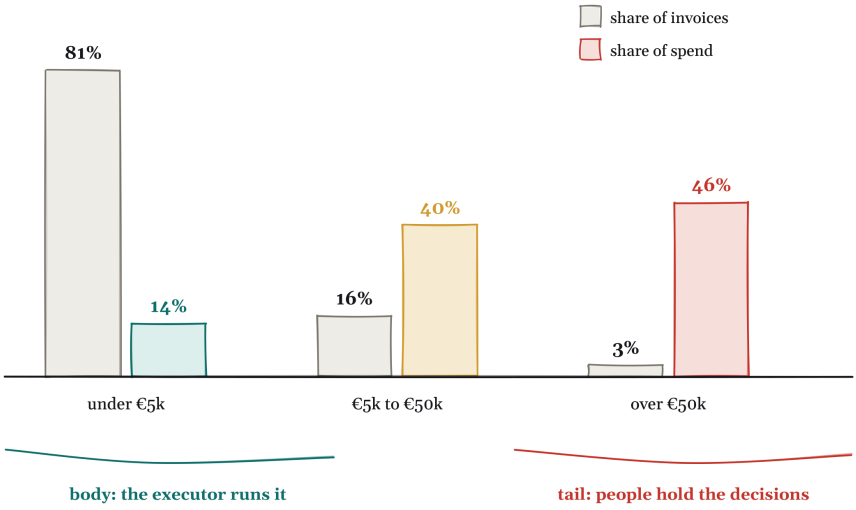
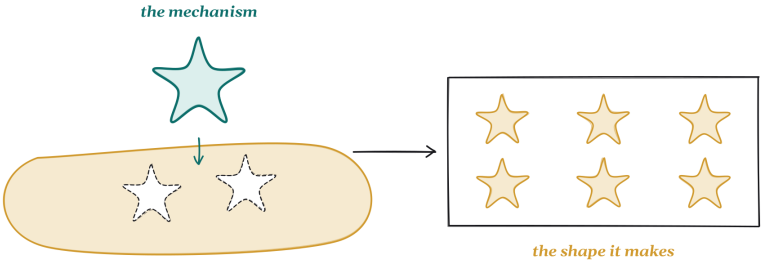


FIGURE 9 Reading the distribution. Share of cases against share of value, by band: a routine body that carries little of the value and a thin tail that carries almost half.

X

How Distributions Are Made

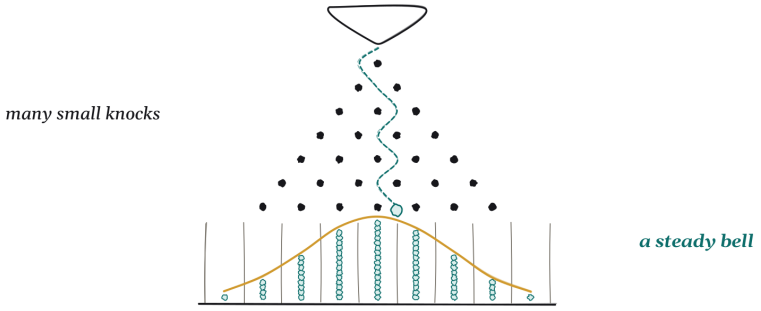
Distributions are not just found. They are produced by the mechanisms that generate the cases, and a business engineer who understands the mechanism can predict the shape, and sometimes change it.



Like a cookie cutter: *the shape of every cookie comes from the cutter, so whoever knows the cutter can predict the cookies, and change them.*

Independent causes make normal shapes

When many small, independent influences add together, the result tends towards a normal distribution. The time a routine form takes depends on many small things, none dominant, and the times cluster. Work like this is stable and predictable, and its average is a good guide.



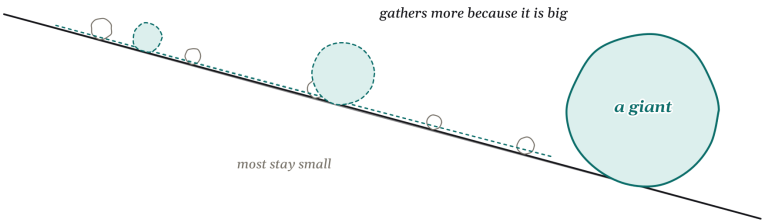
Like a bean machine: *each ball is knocked left or right by many small pegs, none dominant, and together they pile into a steady bell.*

Compounding makes heavy tails

When outcomes feed on themselves, the result is a heavy tail. A customer who buys more gets better terms and buys more again. A product that sells is

stocked more widely and sells more. A website with many links attracts more links. The sociologist Robert Merton called this the Matthew effect, after the line "to every one that hath shall be given". Albert-László Barabási and Réka Albert showed in 1999 how this preferential attachment produces power laws in networks. Wherever there is compounding, there will be a few giants and a long tail of small cases.

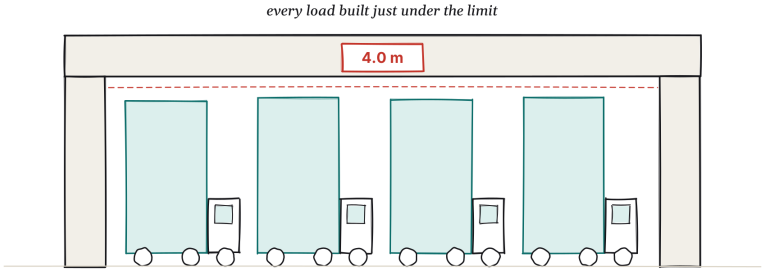
In the platform era, business models were built deliberately on such loops. Network effects, data advantages and distribution flywheels were engineered to compound. Business engineering grew up reading these loops from the outside. It reads them from the inside too: a company's largest customers, suppliers and contracts are heavy-tailed because the relationships compound.



Like snowballs rolling downhill: *each one gathers more snow because it is already big, so a few become giants and most stay small.*

Rules and thresholds reshape distributions

Distributions inside a company are also shaped by its own rules. An approval limit at €5,000 creates a cluster of invoices just below it. A policy that routes every claim over a threshold to a senior adjuster creates a queue at that threshold. A target creates bunching near the target. Many odd shapes in a baseline are the fingerprints of old rules. That is useful, because it shows where the rules are steering the work.

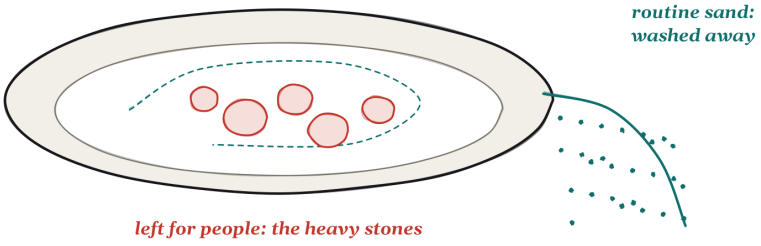


Like lorries under a low bridge: *every load is built just under the posted height, so the bunching is the fingerprint of the rule, not of the cargo.*

Embedded loops make new distributions

Finally, embedding changes distributions. When an executor clears the routine body quickly, the work that remains for people is almost all tail. People's time redistributes towards the hardest cases. Customers and suppliers adapt to faster responses, and the shape of what arrives shifts. This is why the method re-reads the distribution after a loop runs, and why a map review a year later rarely looks like the first.

Understanding how a distribution is made turns reading into prediction. If the cases come from compounding relationships, expect a heavy tail and design for it. If they come from many small independent causes, expect a stable body and trust its average. If they are shaped by a rule, ask whether the rule still serves its purpose.



Like panning for gold: *once the water carries off the fine routine sand, almost everything left in the pan for people is heavy stones.*

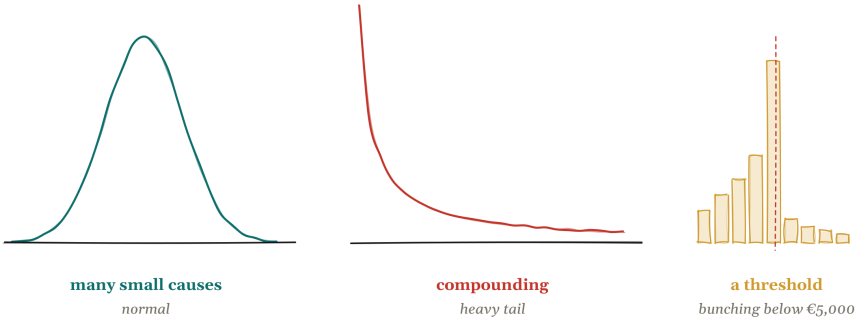
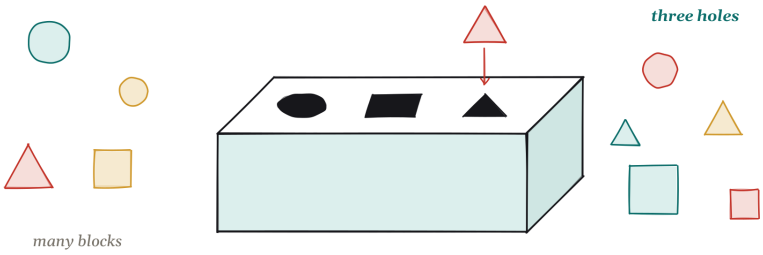


FIGURE 10 How distributions are made. Many small independent causes add up to a normal shape; compounding, where the large grows faster, produces a heavy tail; rules and thresholds leave their own fingerprints.

XI

Abstraction

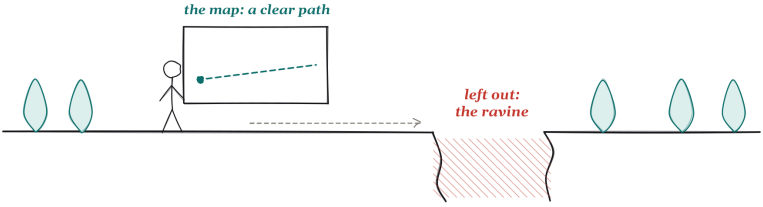
Abstraction is the act of seeing the few kinds of case behind the many. It deliberately loses detail so that the differences that matter become visible.



Like a child's shape sorter: *blocks of every colour and size, yet only three holes, because the shape is the difference that matters.*

The map is not the territory

In 1933, the philosopher Alfred Korzybski put the central warning of abstraction into a phrase that has lasted: "the map is not the territory". Every abstraction leaves things out. That is its purpose, and also its danger. A good abstraction leaves out what does not matter for the decision at hand and keeps what does. A bad one leaves out exactly the thing that will cause trouble.



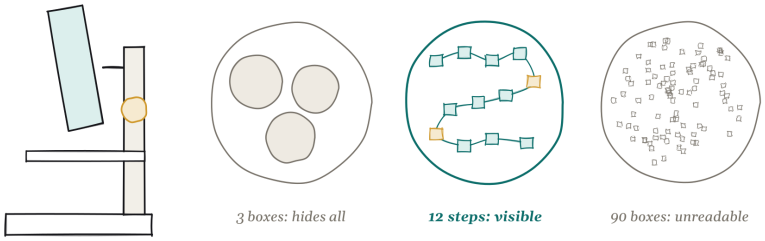
Like a trail map that rightly leaves out every tree, and wrongly leaves out the ravine: *an abstraction is judged by what its omissions cost.*

Choosing the level

The same work can be described at many levels. There are keystrokes and screens, steps and handoffs, decisions and rules, loops and outcomes, and finally lines in the financial model. Each level has its own useful regularities, and a problem that is hopeless at one level can be simple at another. The time an invoice takes is invisible at the level of keystrokes and obvious at the level of

waits between steps. Whether a loop is worth embedding is invisible at the level of steps and obvious at the level of outcomes and economics.

Much of a business engineer's skill is choosing the right level for the question. Process maps that go wrong usually go wrong on level. Three boxes hide everything, and ninety boxes cannot be read. Between eight and twenty steps per loop tends to be the level at which workarounds, waits and decisions become visible.

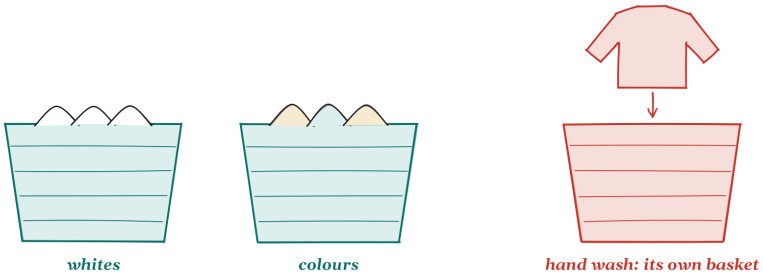


Like a microscope's focus: *three blurred blobs hide everything, ninety specks cannot be read, and in between a dozen steps come sharp.*

Categories that predict

The most useful abstractions are categories whose members behave alike. At the manufacturer, thousands of invoices became three kinds. Small, order-backed invoices from known suppliers were the body. Mid-sized invoices with occasional mismatches came next. Large or unusual invoices needed judgement. Each category predicted how the invoice should be handled. A category that does not predict behaviour is only a label.

Abstraction is where many AI programmes first go wrong. A pilot is built on the cases that were easy to collect. Those cases are abstracted into "invoices", and the tail is quietly left out of the category. The executor then meets the tail in production as a surprise. A good abstraction names the tail as a category of its own from the start.



Like sorting laundry: *each basket predicts its wash, and the delicate jumper needs a basket of its own from the start, or it shrinks.*

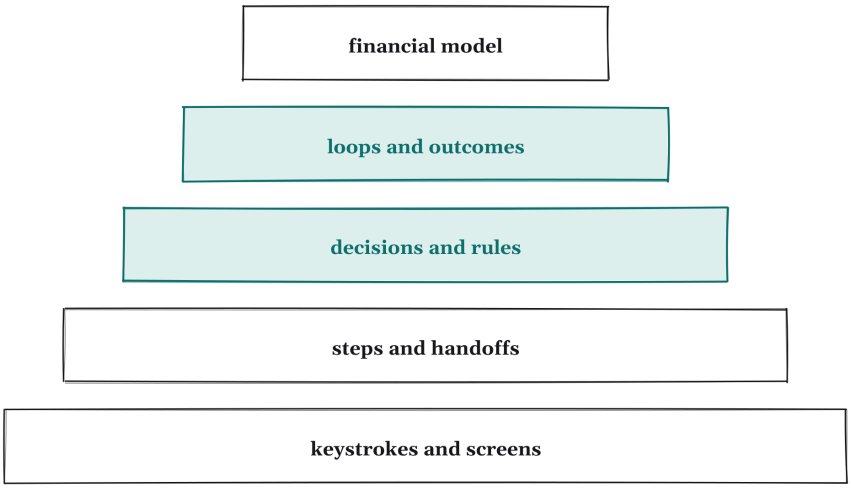
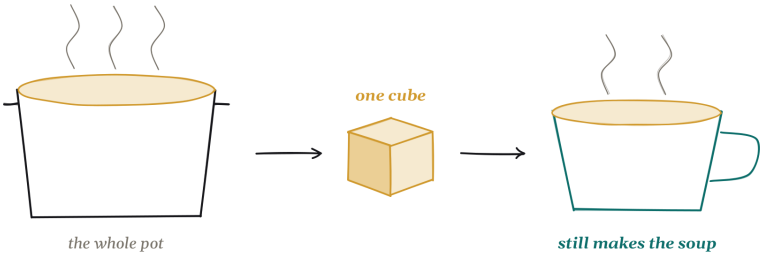


FIGURE 11 Abstraction. The same work at different levels, from keystrokes to the financial model. Each level has its own useful regularities; choosing the level is most of the skill.

XII

Compression

Compression is stating a mechanism in the fewest words that still predict something. If abstraction finds the categories, compression finds the rule that connects them.

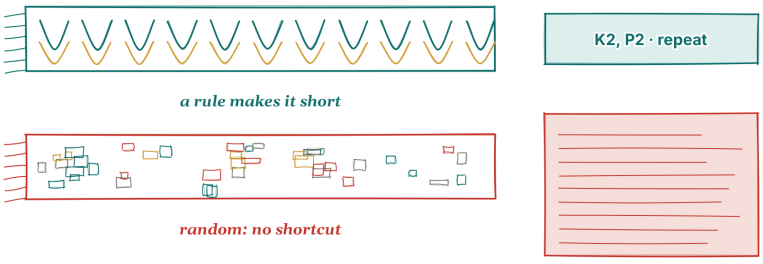


Like a stock cube: *a whole pot of soup reduced to one small cube that still makes the soup again when you add water.*

Prediction and compression

There is a deep link between compressing and understanding. In information theory, data that follows a pattern can be described more briefly than random data, and finding the shortest description means finding the pattern. Mathematicians formalised the idea in the 1960s as the shortest program that reproduces a given output. Statisticians turned it into a principle for choosing models: prefer the model that explains the data in the fewest total bits.

Language models are an instance of the same idea. Trained to predict the next word, they become extraordinarily good compressors of text. Researchers have shown that the two tasks are equivalent. The model is a compression of the distribution it learned from.

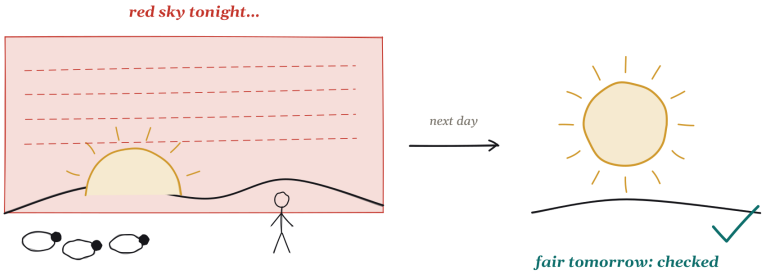


Like a knitting pattern: *a patterned scarf fits on one short line because it follows a rule, while a random one can only be described stitch by stitch.*

Human compression

A business engineer compresses too, but in words that people can use and check. "Most of the elapsed time is waiting between teams" compresses a sixteen-day process into a mechanism. So does "the executor can take the body; people must hold the tail", and so does "a contested decision cannot be engineered away". Each can be tested against new cases and found right or wrong. Each tells someone what to do.

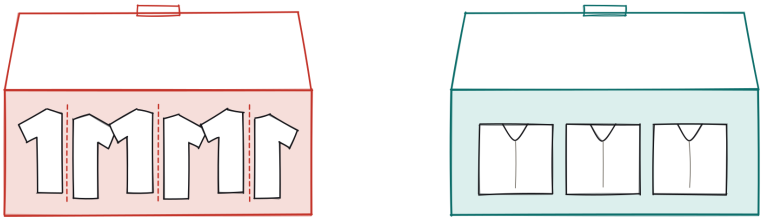
The test of a good compression is that it predicts something you did not already know. "Invoices are complicated" compresses nothing. "Missing goods receipts cause most of the remaining exceptions" predicts that fixing the receiving loop will clear the queue, and it can be checked.



Like the saying 'red sky at night, shepherd's delight': *a few words that predict tomorrow, which tomorrow can prove right or wrong.*

Fewer ideas, given room

Compression is not the same as shortening. A compressed explanation drops the ideas that do not carry weight and gives room to the ones that do. It does not keep every idea and delete the reasoning that connects them. A slide full of fragments is not compressed. It is merely truncated. The discipline of compression is choosing the few mechanisms that matter and stating each clearly enough to be tested.



Like packing a case: *leave out what carries no weight and fold the rest well, rather than cutting every shirt in half to make it all fit.*

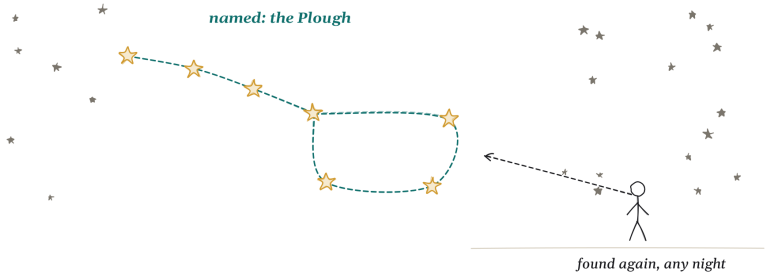


FIGURE 12 Compression. Many cases go in; a short statement that predicts comes out. The test of a good compression is that it can be checked against new cases and found right or wrong.

XIII

Mental Models

A compression that proves useful is given a name, and becomes a mental model. Names matter because they let people find the same pattern again, in a new place, quickly.

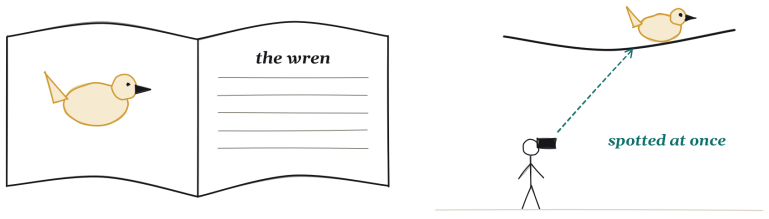


Like a constellation: *once a few stars are joined and named the Plough, anyone can find the same pattern again, quickly, on another night.*

Why naming works

A team that has the words "contested decision" will spot one in a meeting. A team without them will argue about the case in front of it and never notice that the argument has happened twenty times before. Naming turns a one-off insight into a tool that travels. It is also how a discipline becomes teachable: a vocabulary of named patterns that newcomers can learn and practitioners can share.

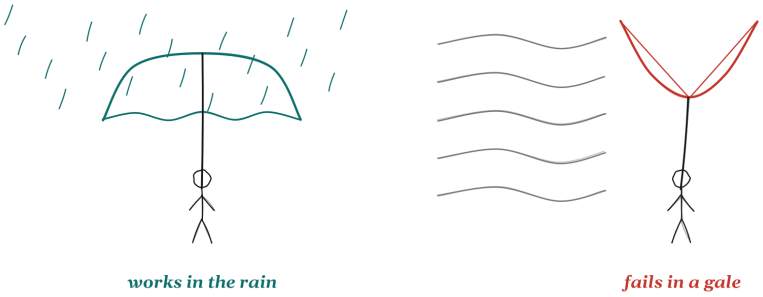
Most entries in this book are named models of this kind. They include the body and the tail, the compensation, the strip ratio and the junction. They also include the four classes of decision, the switch-off, cost per accepted outcome and the Scale Gates. Each compresses a mechanism and gives it a handle.



Like a birdwatcher's field guide: *with the name in hand, the walker spots the bird at once instead of arguing about yet another brown one.*

All models are wrong

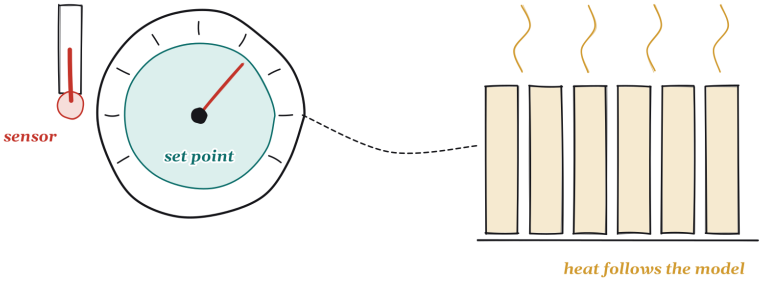
The statistician George Box wrote that all models are wrong, but some are useful. A mental model is a deliberate simplification, and it will fail somewhere. The discipline is to know where. The body-and-tail model fails when a small case is risky. The four decision classes fail when the organisation changes its mind. Good practitioners hold their models firmly enough to act on and loosely enough to revise.



Like an umbrella: *a simple tool that keeps you dry in rain and turns inside out in a gale, so you carry it and watch the wind.*

Models as regulators

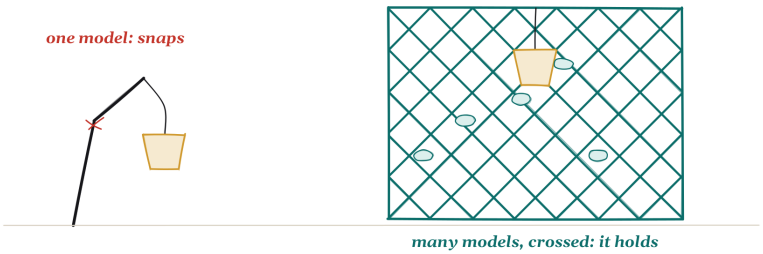
There is a stronger reason models matter in embedding. Conant and Ashby's good regulator theorem says that controlling a system well requires a model of it. In an embedded loop, the company's models do the regulating. The distribution reading, the registers, the limits, the standard and the economics are the means by which people control an executor that decides. Weak models mean weak control. That is why this book treats a register or a standard as part of the machine, not as paperwork about it.



Like a thermostat: *it keeps the room right only because it carries a model of the room, a set point and a sensor; weak model, weak control.*

A latticework

No single model is enough. The investor Charlie Munger described good judgement as a latticework of models from many disciplines, each correcting the others' blind spots. The models in this book are drawn from statistics, logic, cybernetics, psychology, accounting and operations. Their value comes from being used together. Reading a loop through only one of them, whether cost or risk or the technology, produces the one-sided designs that stall.



Like a garden trellis: *a single stick bends and snaps under the load, while many crossed laths hold it because each covers the others' weak spots.*

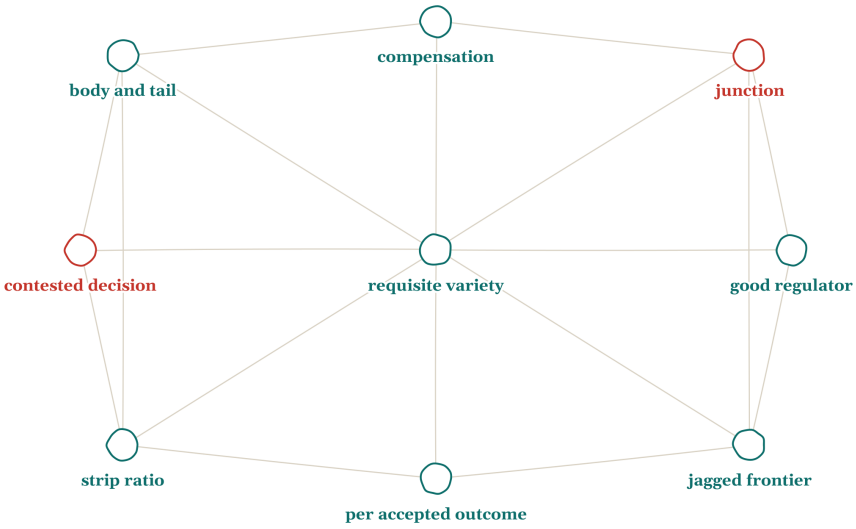
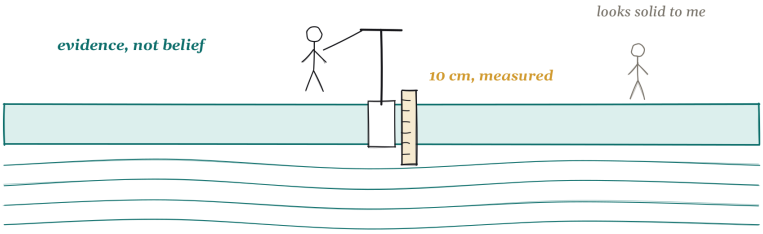


FIGURE 13 Mental models as a latticework. Named compressions from many fields, used together, each correcting the others' blind spots.

XIV

Testing and Judgement

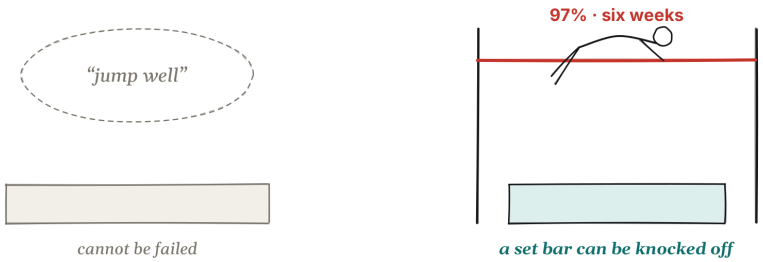
Every model is a claim about the world, and claims must be tested. Testing replaces belief with evidence. Over many cycles, it builds the judgement that lets a practitioner act well when evidence is thin.



Like drilling the ice before walking on the lake: *belief says it looks solid, the drill measures how thick it really is.*

Falsifiable claims

The philosopher Karl Popper argued that a claim is scientific only if some observation could prove it false. The same standard works for business models. "The executor will improve the invoice process" cannot be proven false. "The executor will match the clerks' results on at least 97% of invoices under €5,000 over six weeks" can. Shadow running, gate reviews and measured economics are all ways of putting falsifiable claims to the test before relying on them.

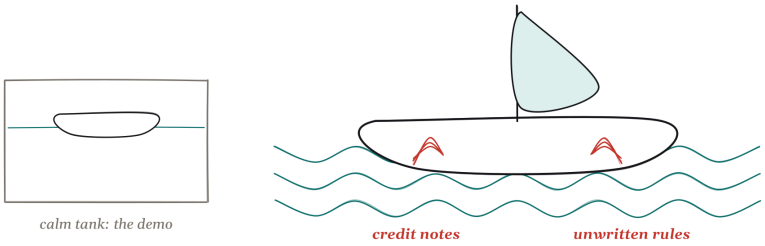


Like a high-jump bar: *'jump well' can never be failed, but a bar set at a stated height can be knocked off, so clearing it means something.*

Testing on real cases

Tests must use real cases, including the tail. A demonstration on prepared cases tests the demonstration. Shadow running on the live stream of cases

tests the loop. It reveals what the abstraction left out: credit notes that look like invoices, remittances in unfamiliar formats, rules the clerks apply that nobody wrote down. Every one of those findings improves the model.



Like a boat's sea trial: *a calm test tank proves only the tank, while the open sea finds the leaks nobody knew were there.*

Where judgement can be trusted

Testing builds judgement: a practitioner's sense of what will work, earned from being right and wrong many times. Research by Daniel Kahneman and Gary Klein suggests that expert intuition is reliable only in environments that are regular and give fast, clear feedback. It is unreliable where cases are rare, varied and slow to show whether a decision was right.

The body of most processes is the first kind of environment, and the tail is often the second. The same expert who can be trusted to say "that invoice is fine" in seconds should be asked to show evidence on a large disputed one. The same holds for an executor. High agreement on the body says little about its judgement on the tail.

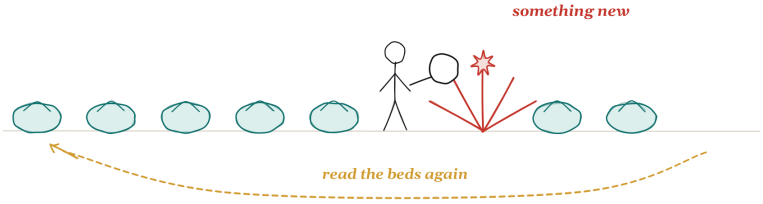


Like a darts player: *trust the throw at the board practised on every day, not at a far board in the fog that is rarely seen.*

Judgement notices change

Judgement does one more thing that tests cannot. It notices when the world has changed: a new kind of case, a shift in the distribution, a supplier that

redesigned its invoices. That noticing sends the method back to its first move. The practitioner reads the distribution again, and the cycle restarts.



Like a gardener who spots a plant nobody sowed: *noticing something new sends the work back to reading the beds again from the start.*

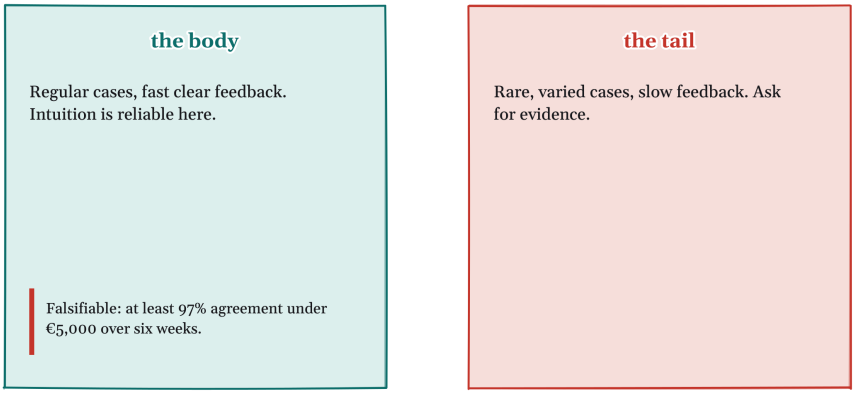
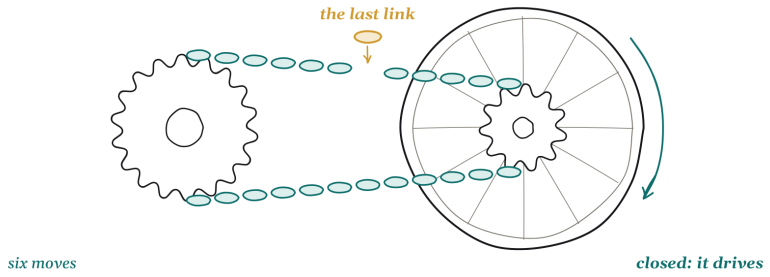


FIGURE 14 Testing and judgement. Falsifiable claims are put to real cases; intuition is trusted where feedback is fast and regular, in the body, and asked for evidence where it is not, in the tail.

XV

The Closed Loop of Business Engineering

The six moves of the method now close into a loop, and the loop connects to real action. This is what turns business engineering from a way of understanding a company into a way of running one.

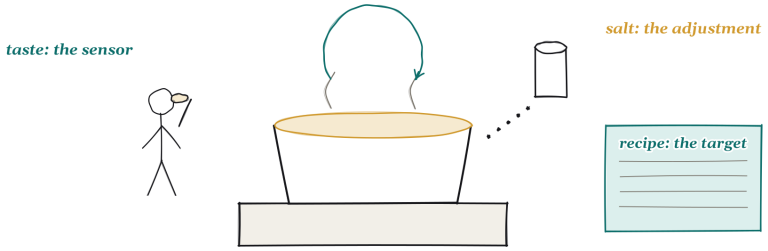


Like joining the last link of a bicycle chain: *the six links of the method only drive the wheel once the loop is closed.*

From understanding to control

Reading the distribution, abstracting, compressing and naming produce a model of the work. Testing anchors the model in reality, and judgement keeps it current. In cybernetic terms, that model is the regulator. Each loop's registers, limits, junctions and standard are how the company controls the executor that does the work. The grading of every outcome is the sensor. The operating review is the comparator, and the decisions taken there are the adjustment.

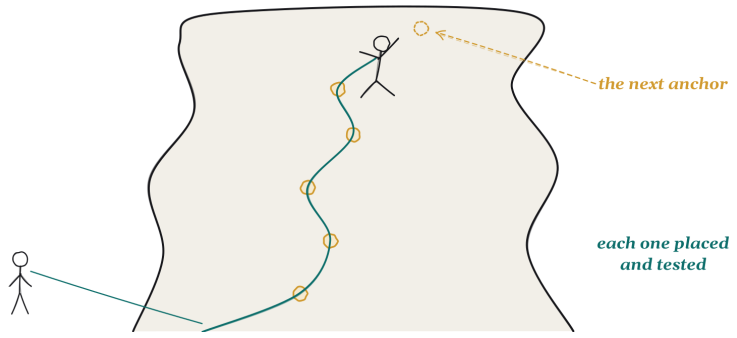
A loop built this way is closed. It acts on cases, senses whether the outcomes meet the standard, compares them with targets and adjusts. A loop without grading, without review or without a named owner is open. It acts and hopes.



Like a cook who tastes the soup: *act, taste, compare with the recipe, adjust the salt; a cook who never tastes simply acts and hopes.*

From one loop to many

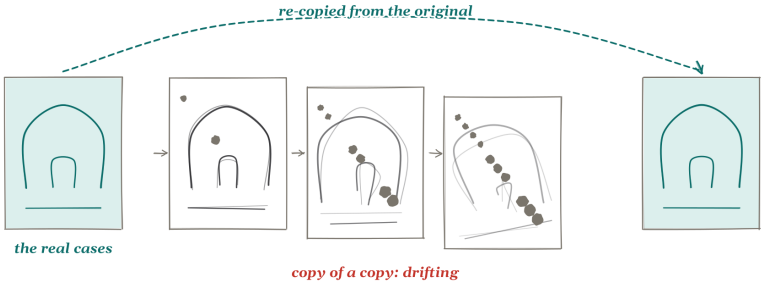
The same structure repeats at a larger scale. Each loop in the core feeds the map review: what its exceptions say about upstream loops, where people's time now sits, how readiness has changed. The map review chooses the next loop to read. The company's model of itself grows one loop at a time, and each cycle starts from a better reading than the last.



Like a climber placing anchors: *each one placed and tested makes the next pitch safer, so the route grows one secure step at a time.*

The strange loop, kept honest

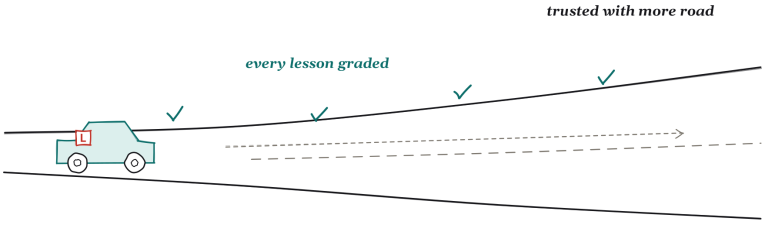
Here the strange loop from Part One becomes practical. The company builds a model of its work and acts through it, and the action changes the work. The changed work then changes the model. This reflexivity is not a flaw. It is how a company learns. But it can drift into fiction if the model stops being anchored in real cases. Registers can be copied from manuals rather than walked from cases. Standards can grade activity rather than outcomes. Reviews can stop reading the tail. The disciplines of the method are what keep the loop honest: read the real distribution, measure rather than estimate, test on real cases, and get decisions signed.



Like a photocopy of a photocopy: *each copy of the model drifts further from life unless the next one is taken again from the real original.*

Real-world action and interaction

Closed loops are what make AI useful in the world rather than in a demo. An executor that acts inside a closed loop can be trusted with more over time, because every action is graded and every error is fed back. One that acts in an open loop cannot, however capable it is. The core of business engineering, in the end, is this: build models of the work good enough to regulate it, close the loop between action and evidence, and keep the model honest as the world changes. The nine areas that follow are where that core is applied.



Like a learner driver: *because every lesson is graded and every mistake corrected, the learner is trusted with busier roads over time.*

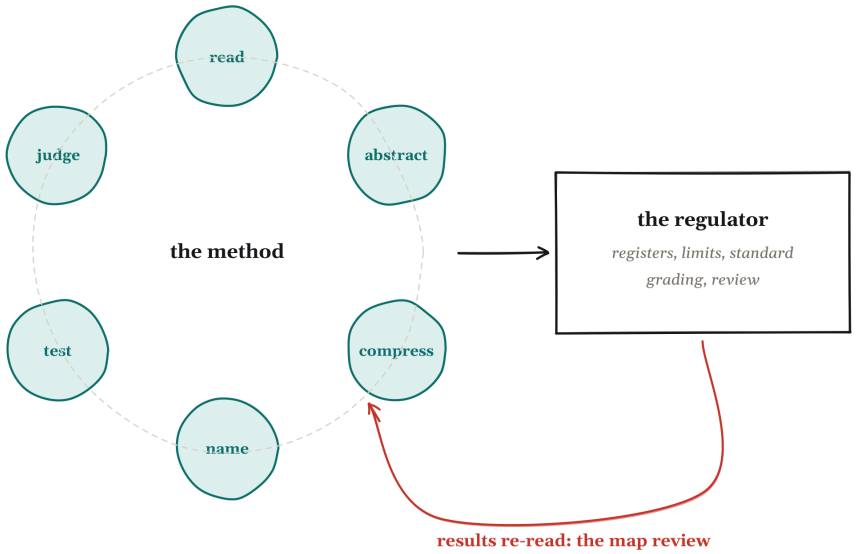


FIGURE 15 The closed loop of business engineering. The method builds the model that regulates each loop; each loop's results feed the map review; the map review chooses the next loop to read.

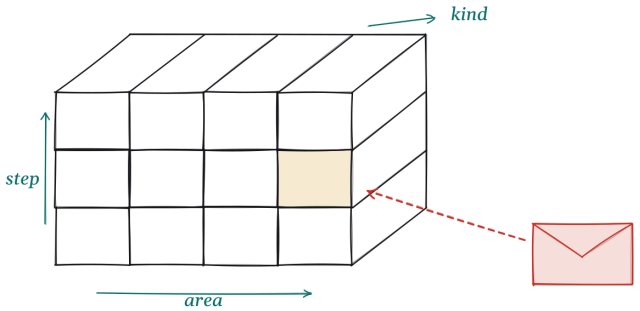
PART THREE

Reading the Business

XVI

The Shape of This Body of Knowledge

A body of knowledge needs an architecture, so that a reader can find any concept and see how it relates to the others. This one has three dimensions.



Like a post-office sorting cabinet: *every concept lands in one pigeonhole found by three coordinates, its area, its step in the lifecycle and its kind.*

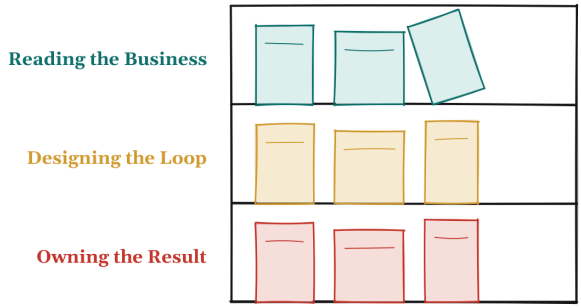
Nine areas

The first dimension is the kind of knowledge. There are nine areas, grouped into three parts.

Part Three, Reading the Business, holds three areas. Distributions covers how cases spread and what that implies. Value and the Map covers which loops matter and in what order. Work and Process covers how work actually runs.

Part Four, Designing the Loop, holds three more. Rules and Decisions covers what the loop must follow and who decides what. The Executor covers what does the work and what it is made of. Control and Acceptance covers how anyone knows the work is good and how it is stopped when it is not.

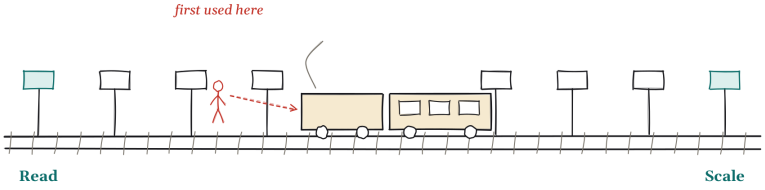
Part Five, Owning the Result, holds the last three. People and Accountability covers who answers for what. Economics covers what the work costs and what it is worth. Governance covers how decisions are recorded and how they meet the company's obligations.



Like a bookcase with three shelves of three volumes: *reading the business on top, designing the loop in the middle, owning the result below.*

Ten steps

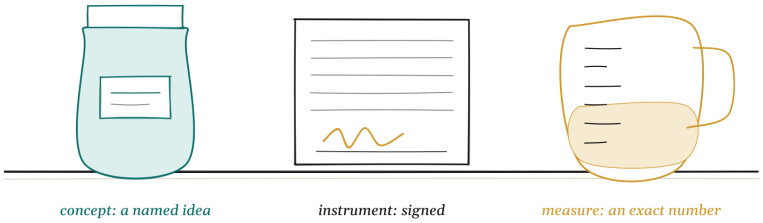
The second dimension is time. Every concept has a place on the ten-step embedding lifecycle: Read, Map, Strip, Sort, Design, Shadow, Bound, Own, Core and Scale. Each area opens with a table showing where its concepts are first used. Appendix A gives the full matrix of areas against steps.



Like a railway line with ten stations from Read to Scale: *every concept has a stop where it first boards the train.*

Three kinds of content

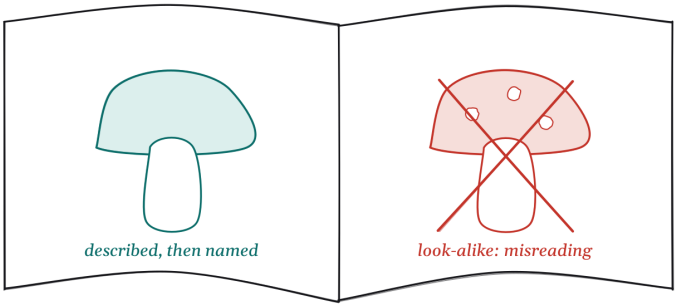
The third dimension is the kind of content. Each area contains concepts, which are ideas with names; instruments, which are documents or drawings a team fills in and signs; and measures, which are numbers with exact definitions. The entries say which is which. Part Six gathers the notation for the instruments and the formulas for the measures, so they are defined in one place.



Like a kitchen shelf: *a labelled jar is a concept, a signed recipe card is an instrument, and a marked measuring jug is a measure.*

How entries are written

Every entry follows the same discipline, though not the same template. It defines the concept in plain words before naming it. It says why the concept matters, usually by describing what goes wrong without it. It shows how the concept is used, often with the composite manufacturer. And it names the common misreadings, because in a young discipline the misreadings spread faster than the definitions.



Like a mushroom field guide: *each entry describes the thing plainly, then names it, then warns about the look-alike that people mistake for it.*

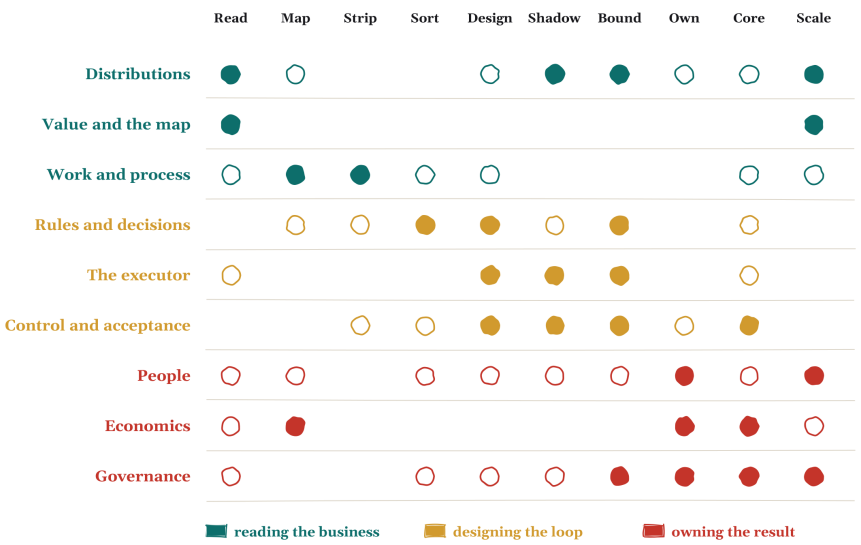
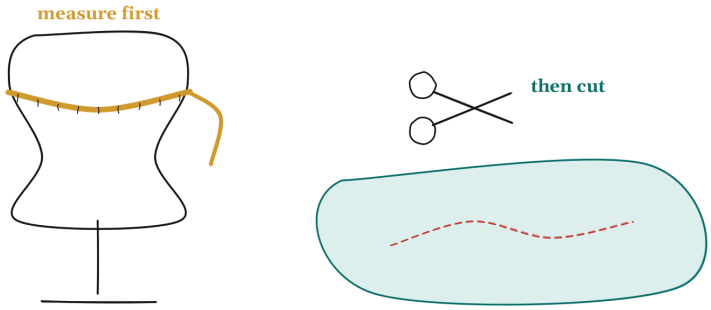


FIGURE 16 The shape of this body of knowledge. Nine areas of knowledge read against the ten steps of the embedding lifecycle; filled marks show where an area is central, open marks where it contributes.

XVII

Distributions

This area covers how the cases a business handles are spread, and what that spread implies for design. It comes first because every later decision depends on it. Which loop to start with, what an executor may do alone, where people must decide, how long to test and how to report results all turn on the shape of the cases.



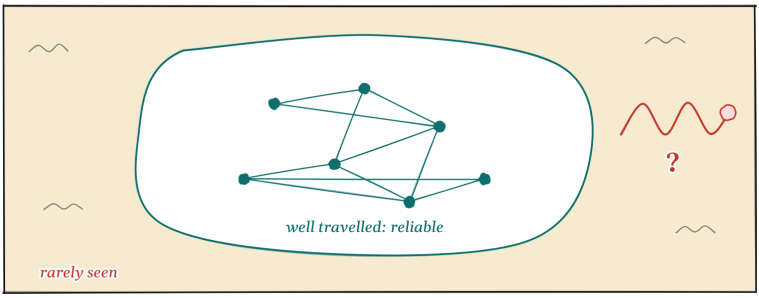
Like a tailor who measures before cutting: *the shape of the cases decides where to start, what runs alone, where people decide and how long to test.*

Where it comes from

This area applies the first move of the method. Its foundations are statistics and the science of complex systems: normal and heavy-tailed shapes, and the mechanisms that produce them. In AI terms, a learned model is a compression of the distribution it was trained on. So the shape of a loop's distribution predicts where the model will be reliable and where it will not.

Concept	Kind	First used at
Case	Concept	Read
Distribution	Concept	Read
Normal and heavy-tailed shapes	Concept	Read
Body and tail	Concept	Read
Share of cases against share of value	Measure	Read
Segmenting by value, difficulty and risk	Concept	Read, Shadow
Distribution shift	Concept	Core, Scale

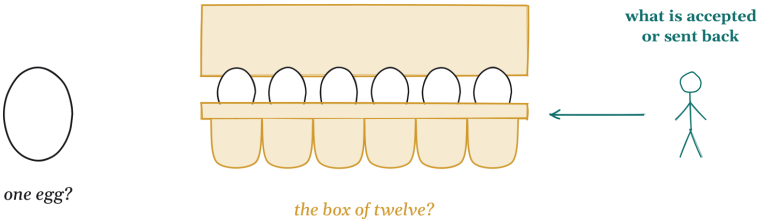
Concept	Kind	First used at
Where intuition can be trusted	Concept	Shadow, Core



Like an old map drawn from travellers' reports: *sharp where many people went, blank where few did, just as a model is reliable only where its training cases were dense.*

Case

A case is one unit of work that a loop handles from start to finish: one invoice, one claim, one support ticket, one job application. Defining the case is the first act of reading a loop, and it is less obvious than it sounds. Is the case one invoice, or one supplier's monthly batch? Is it one ticket, or one customer's problem across three tickets? The right answer is the unit whose outcome a receiver can accept or reject. Getting it wrong distorts every count that follows.



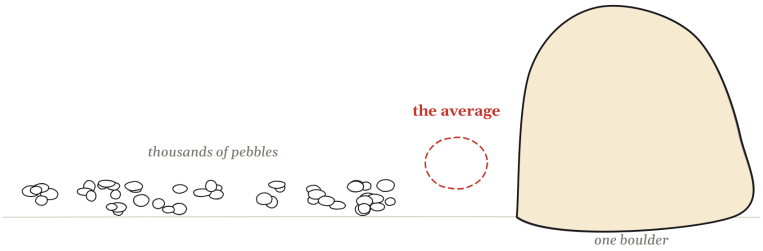
Like a grocer choosing the unit of sale, one egg or the box: *the right case is the unit a receiver can accept or send back.*

Distribution

The distribution of a loop is how its cases spread along some dimension: their value, their difficulty, their risk, or the time they take. It is read by counting cases in bands and comparing the bands. A distribution answers a simple

question that most companies cannot answer about their own processes: what do our cases actually look like?

A common misreading is to treat the average case as representative. In many loops no actual case looks like the average. A loop with thousands of tiny invoices and a few enormous ones has an average invoice that hardly anyone ever sees.



Like the average stone on a beach of pebbles and one boulder: *it describes no stone that is actually there, just as the average case may not exist.*

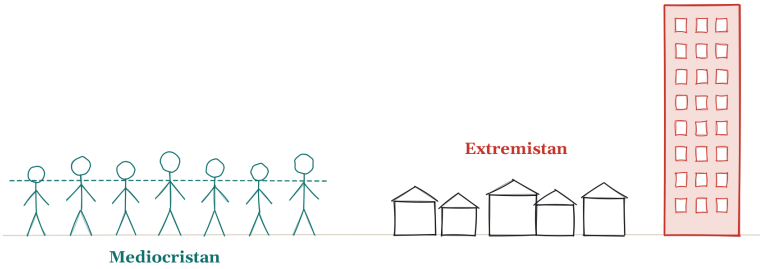
Normal and heavy-tailed shapes

Distributions come in two broad shapes, and most real loops are a mixture.

In a roughly normal distribution, cases cluster around a typical value and extreme cases are rare and not very extreme. Weekly hours on a timesheet behave like this, and so do routine address changes. In these loops, averages are meaningful, rules work well, and a sample tells you a lot about the whole.

In a heavy-tailed distribution, a small share of cases carries most of the value or the risk, and the largest cases can be enormous relative to the typical one. Invoice values, insurance claims, customer revenue and contract sizes often behave like this. The commonest heavy-tailed shape is called a power law. In these loops, averages mislead, a sample can miss the cases that matter most, and the design has to treat the tail differently from the body. The essayist Nassim Taleb popularised a pair of names for these two worlds: "Mediocristan" for the first and "Extremistan" for the second.

The practical test is simple. Rank the cases by value and ask what share of the total the top few percent carry. If the top 5% carry 10% of the value, the loop is close to normal. If they carry half, it is heavy-tailed.



Like people's heights against the buildings of a city: *heights cluster near a typical value, while one tower can dwarf a whole street of houses.*

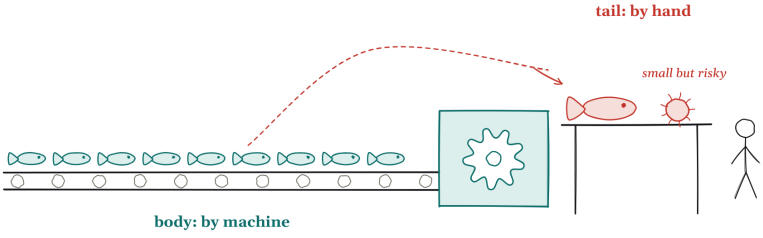
Body and tail

Most real loops have two parts. The body is the large share of routine cases that resemble each other. The tail is the small share of cases that carry most of the value, the risk or the difficulty.

The distinction is the most useful single idea in embedding. An executor that decides can usually run the body, because its cases repeat and feedback is fast. People should usually hold the tail, because the cost of error is high and each case is different. Many design choices follow from it. Junctions sit mostly in the tail. Shadow results are reported separately for each part, and so are the economics. And the first bounded scope usually covers the body.

At the manufacturer, 81% of supplier invoices were under €5,000 and carried 14% of the spend. That was the body. The 3% over €50,000 carried 46% of the spend. That was the tail.

A common misreading is to draw the line between body and tail by volume alone. A case can be small in value and still belong in the tail because it is risky, such as a first invoice from a new supplier with new bank details.

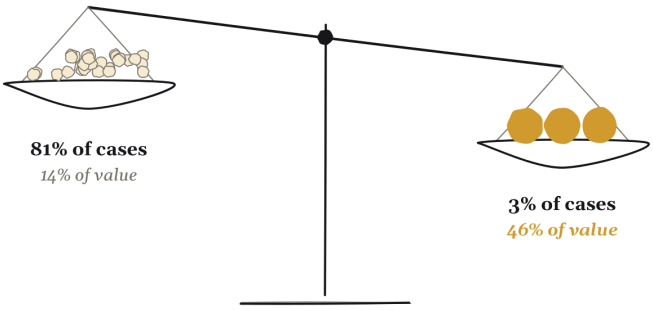


Like a fish market: *the many ordinary fish run through the machine, while big catches and small poisonous ones go to a person's table.*

Share of cases against share of value

The basic measure of a distribution compares, band by band, the share of cases with the share of value. It is drawn as paired bars, one for each share, for each band. Where the bars are similar in height, the loop is close to normal. Where the value bar dwarfs the case bar in the top band, the tail is heavy.

The measure is easy to compute from a few months of data. Its definition is given in Part Six. It should be read at Step 1, before a loop is chosen, and again at Step 10, after the body has been embedded.

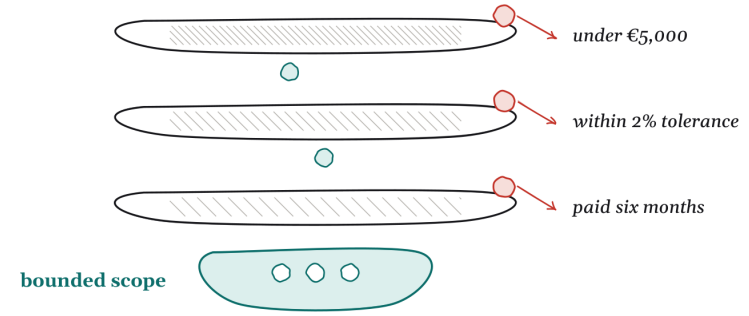


Like a set of scales with a heap of copper coins on one pan and three gold coins on the other: *the few gold coins outweigh the heap.*

Segmenting by value, difficulty and risk

A loop has more than one distribution. Its cases spread differently by value, by difficulty and by risk, and the three do not always line up. A large invoice from a long-standing supplier with a perfect match is high in value but low in difficulty. A small invoice for a new supplier with an unusual tax treatment is low in value but high in difficulty and risk.

Good designs read all three. Bounded scope is usually defined on several dimensions at once, as at the manufacturer: under €5,000, within a 2% tolerance, from a supplier paid for at least six months. Each condition cuts a different distribution.

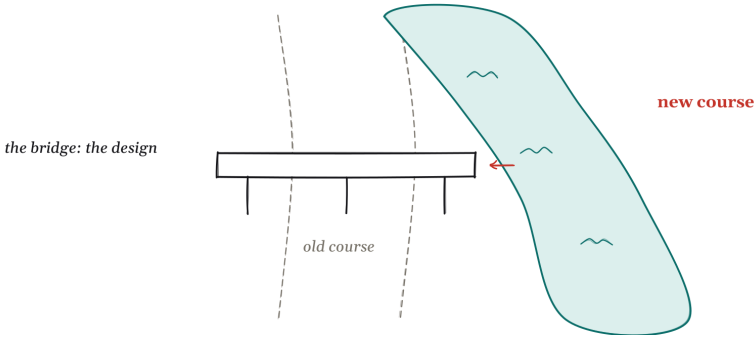


Like three stacked sieves with different meshes: *value, difficulty and risk each catch different cases, and only those passing all three enter the bounded scope.*

Distribution shift

Distributions change. A new product line brings new kinds of case. A regulation changes which cases are risky. A large customer changes how it orders. A shift can quietly move cases from the body to the tail, where the executor should not be deciding alone.

Shifts are caught in two ways. The first is grading every outcome and watching the share that meets the standard, which falls when the executor meets cases it handles badly. The second is re-reading the distribution itself at each map review. Both are needed. Grading catches the effect, and re-reading explains the cause.



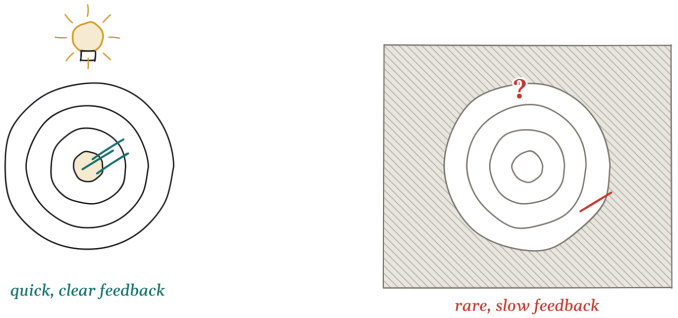
Like a river that changes course after a storm: *the bridge still stands, but the water now runs past its end, and only re-reading the river shows why.*

Where intuition can be trusted

Practitioners develop a feel for which cases are routine and which are not, and that feel guides many small decisions. Research by Daniel Kahneman and Gary

Klein suggests that expert intuition is reliable only where the environment is regular and gives quick, clear feedback. It is unreliable where cases are rare, varied and slow to reveal whether a decision was right.

The body of most loops is the first kind of environment, and the tail is often the second. So the same expert who can be trusted to say "that invoice is fine" in seconds should be asked to show evidence on a large disputed one. The same logic applies to the executor. A high agreement rate on the body says little about its judgement on the tail.



Like a darts player who sees every throw land in a lit room: *skill is trustworthy where feedback is quick and clear, not in the dark.*

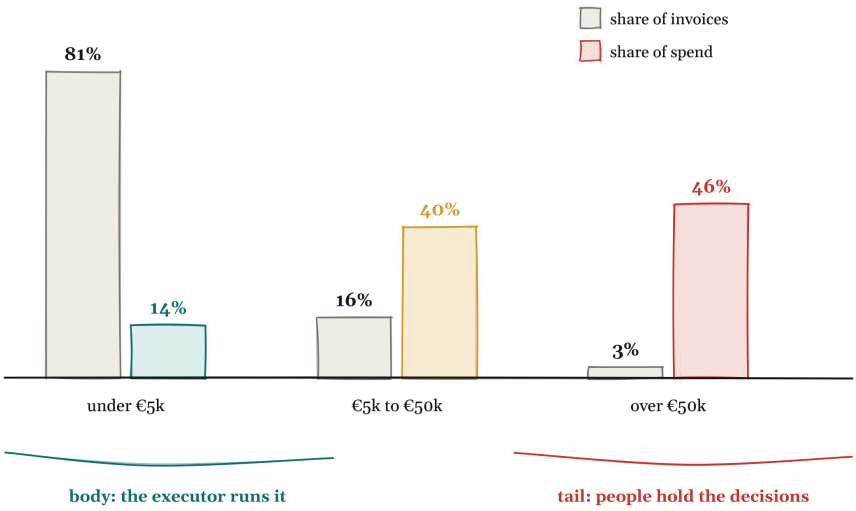
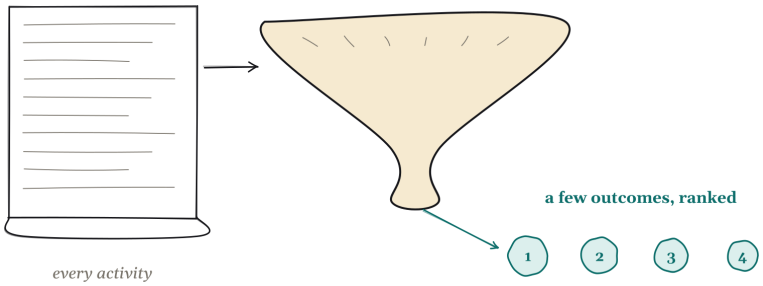


FIGURE 17 Distributions. Share of invoices against share of spend, by band, at the composite manufacturer. The routine body carries little of the value; the thin tail carries almost half.

XVIII

Value and the Map

This area covers how a company decides which work matters, which loop to embed first, and how its loops connect. It turns a company's long list of activities into a small number of outcomes that can be compared, ordered and owned.



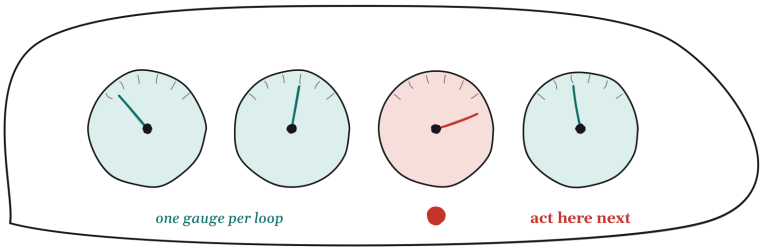
Like a funnel: *the long list of a company's activities goes in, and a few outcomes come out that can be compared, ranked and owned.*

Where it comes from

This area applies abstraction at the level of the whole company. It reads the business from the inside, loop by loop, and connects that view to the business model read from the outside. In cybernetic terms, the map is the company's model of its own work at the highest level: the regulator for deciding where to act next.

Concept	Kind	First used at
Work loop	Concept	Read
Outcome	Concept	Read
Value to the core	Measure	Read
Readiness	Measure	Read
Work Loop Map	Instrument	Read
The four regions of the map	Concept	Read
The first loop	Concept	Read
Chains and handoffs between loops	Concept	Read, Scale

Concept	Kind	First used at
The map review	Instrument	Scale
The inside-out complement to the business model	Concept	Read



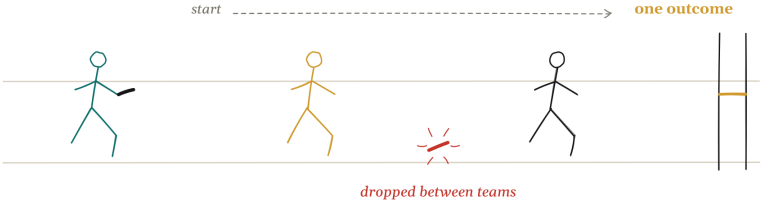
Like a car's dashboard: *built from readings taken inside the engine, it is the driver's model of the machine and tells them where to act next.*

Work loop

A work loop is the unit of work that produces one outcome end to end, with everything needed to produce it well. That means its rules, context, actions, decisions, junctions, exceptions, handoffs, standard, people and economics. It is smaller than a department and larger than a task. It usually crosses teams.

The loop is the unit of embedding. AI is not embedded into "finance" or "customer service". It is embedded into a loop, such as supplier invoice approval or support ticket resolution, one at a time.

A common misreading is to equate a loop with a team's process. A team's process is often a fragment of a loop, and a loop often includes steps no single team owns. That gap is where accountability gets lost.

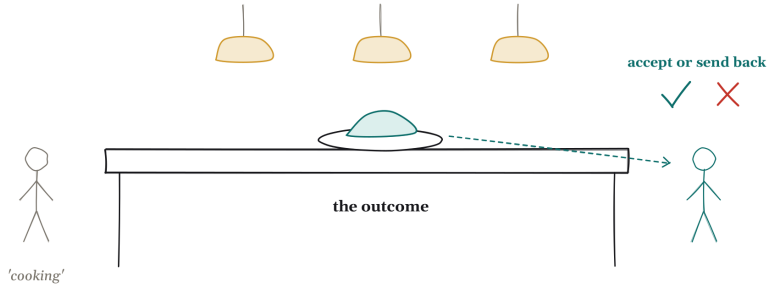


Like a relay race: *the loop is the whole race to the finish, run across several teams, and the baton is usually dropped at a handoff.*

Outcome

The outcome is the result a loop produces, stated so that someone outside the team can tell an accepted result from a rejected one. "An approved supplier invoice, matched and posted for payment on time" is an outcome. "Process invoices" is an activity.

Stating the outcome is harder than it sounds and more useful than it looks. It names the receiver of the work. It also exposes loops whose teams disagree about what they produce, and it gives the standard something to grade.

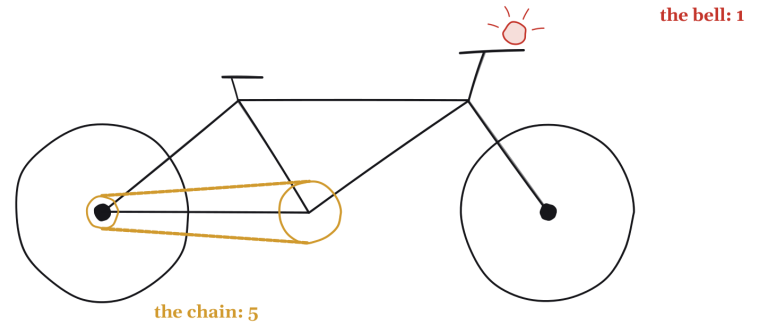


Like a dish placed on the kitchen pass: *the waiter can take it or send it back, whereas 'cooking' is only an activity nobody can accept.*

Value to the core

Value to the core scores how much a loop matters to the business, from one to five. It combines volume, cost, effect on customers, effect on cash and exposure to risk. The score is a judgement, not a calculation, made by the people who know the work and argued over in the room.

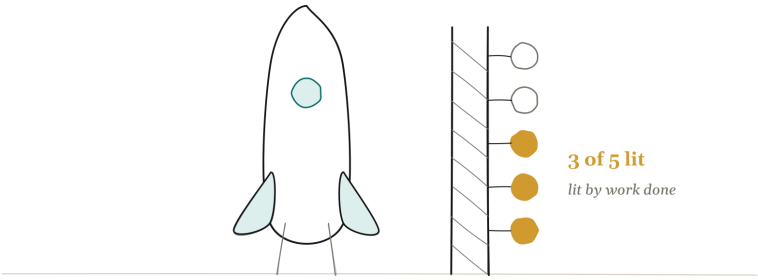
A common misreading is to score value by how impressive the AI would look. Some of the highest-value loops are dull. Supplier invoices, goods receipts and payment runs rarely appear in demos and often carry more value than the customer chatbot.



Like a bicycle's chain and its bell: *the shiny bell draws the eye, but the dull chain carries the ride, and value is scored on the chain.*

Readiness

Readiness scores how close a loop is to being embedded, also from one to five. It asks how well the loop's rules are known, how measurable its outcome is, how much of its context is already in systems, how willing its owner is, and how many unsettled disagreements it contains. Readiness rises as work is done, both on the loop itself and on its neighbours.



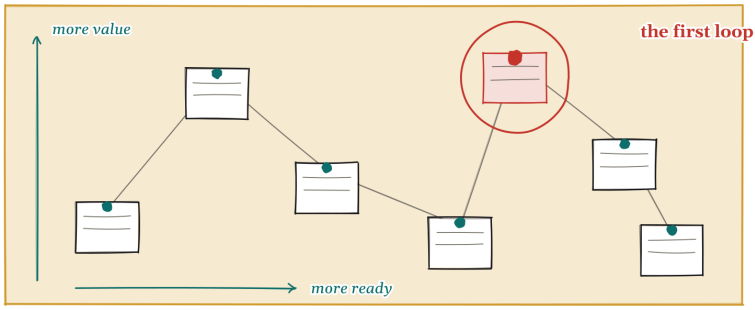
Like a rocket on the launch pad with a row of check lamps: *readiness is how many are lit, and each one is lit by work done.*

Work Loop Map

The Work Loop Map places every loop of a company on two axes: value to the core and readiness. Lines between loops show where one hands work to another. The map is the instrument of Step 1 and Step 10. It is drawn in a working session with the heads of the main functions, and signed by the executive sponsor with the first loop marked.

The map's main use is to make choices visible. Leadership teams rarely compare their processes side by side. The map forces the comparison, and the

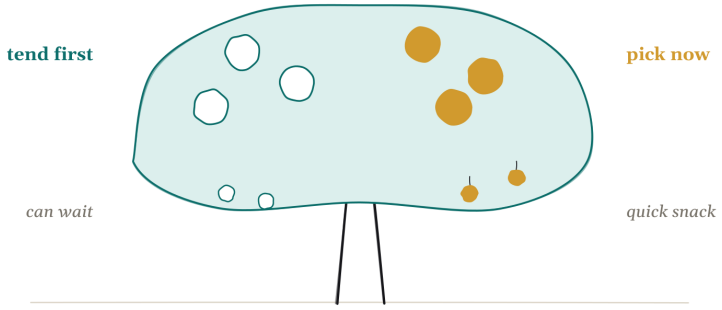
argument over scores is often the first honest conversation about which work matters.



Like a corkboard with every case pinned up and strung together: *once all the loops hang side by side, the room has to compare them and pick one.*

The four regions of the map

The map divides into four regions. High value and high readiness is where to start. High value and low readiness is where to map before building: these loops matter, but their rules or data are not ready, and the lifecycle's early steps are the work to do first. Low value and high readiness holds quick wins, which teach little and save little. Low value and low readiness can wait.

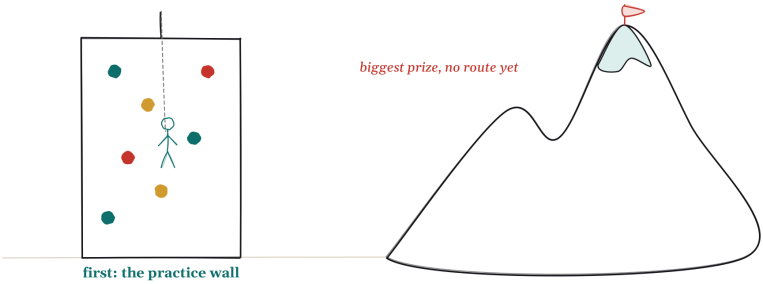


Like an orchard tree: *pick the high ripe fruit now, tend the high unripe fruit first, take low ripe ones as snacks, and leave the rest.*

The first loop

The first loop has a job beyond its own savings: it teaches the company the lifecycle. It should have repeating cases, so evidence accumulates quickly. Feedback should be fast, its outcome measurable, its owner willing, and its distribution should have a body and a tail. The most valuable loop on the map is often not the right first loop, because its readiness is low.

At the manufacturer, order fulfilment scored highest on value, but its rules were scattered across three plants. Supplier invoice approval scored four on value and three on readiness, and it went first.

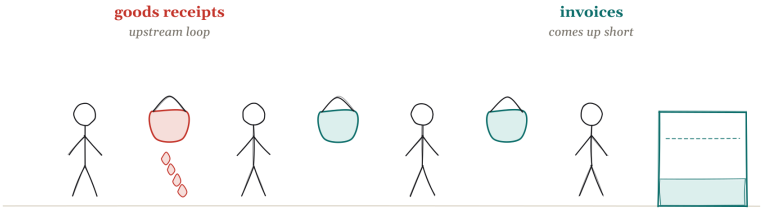


Like learning to climb on the practice wall before the summit: *the first loop is chosen to teach the method fast, not because it is the biggest prize.*

Chains and handoffs between loops

Loops connect into chains, sequences in which one loop's outcome is the next loop's input. Common chains include lead to cash, procure to pay, hire to retire, record to report, issue to resolution and order to delivery. The map draws the handoffs between loops as lines.

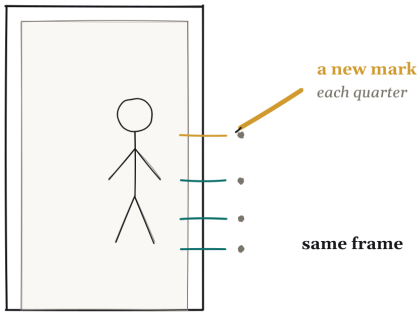
Chains matter because a loop's performance depends on its neighbours. Once the manufacturer's invoice loop ran in the core, 38% of its remaining exceptions came from missing goods receipts, a problem in the loop upstream. Loops in the same chain often can only scale together.



Like a bucket brigade: *each loop passes its output to the next, so a leaking bucket upstream leaves the next loop short however well it works.*

The map review

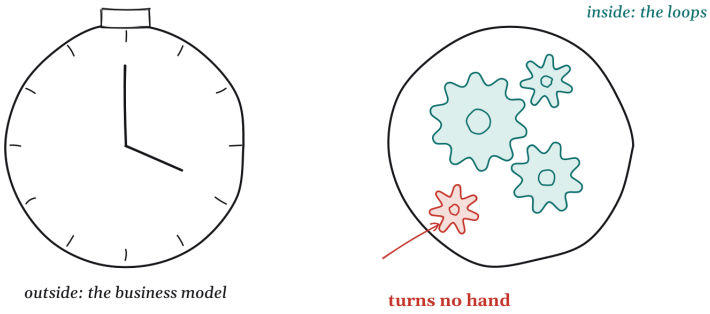
The map review is the quarterly update of the map, signed by the sponsor at Step 10. It updates readiness, records the state of each lifecycle in progress, reads the edges of embedded loops for the next candidates, and chooses what to read next. It is the same instrument that started the lifecycle, one cycle later.



Like height marks on a doorframe: *the same frame, read again each quarter, records how far each loop has grown and where to look next.*

The inside-out complement to the business model

The map is the inside-out complement to the business model. A business model describes what a company is and how it wins, read from the outside: its value, technology, distribution and financial models. The map describes how its work runs, read from the inside. Each loop's outcome should trace to a value the business creates. Each loop's economics should roll up into the financial model. A loop whose outcome traces to nothing the business values is a candidate for removal, not for embedding.



Like a watch's face and its movement: *the face is read from outside, the gears run inside, and a gear that turns no hand should come out.*

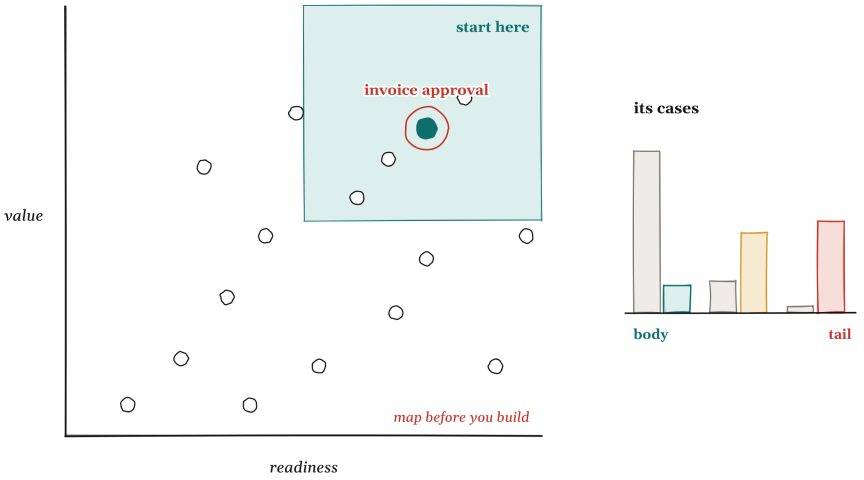
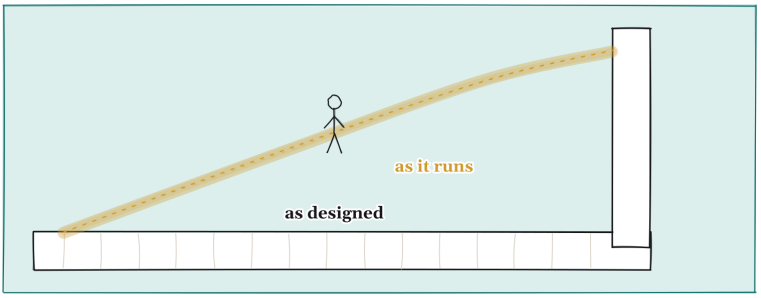


FIGURE 18 Value and the map. Loops placed by value to the core and readiness; the first loop chosen in the region where both are high, with its distribution read before anything is built.

XIX

Work and Process

This area covers how work actually runs inside a loop: its steps, its waits, its handoffs, its workarounds and its costs. It is the knowledge most companies believe they have and most do not, because their process documents describe how work was designed rather than how it runs.



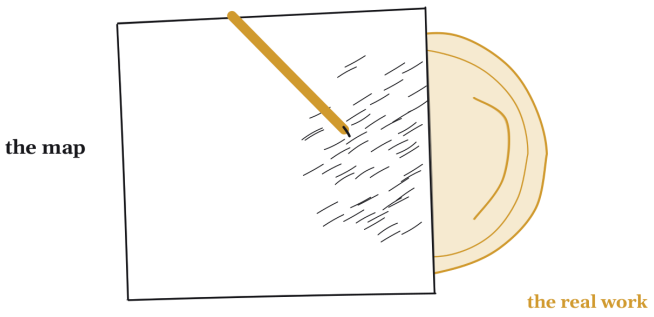
Like the paved path across a lawn and the worn track beside it:
documents show how the work was designed, the worn track shows how it runs.

Where it comes from

This area anchors the symbols in the world. A process map is only meaningful if it mirrors how the work really runs, which is why it is drawn with operators and walked with real cases. Its compensations are the accumulated rules of an older executor, kept long after the constraints that justified them.

Concept	Kind	First used at
Process map	Instrument	Map
Walk-through	Concept	Map
Step, wait and handoff	Concept	Map
Flow efficiency	Measure	Map, Core
Baseline	Instrument	Map
Compensation	Concept	Strip
Strip ratio	Measure	Strip
Topology and accretion	Concept	Strip
Rework	Concept	Map, Strip

Concept	Kind	First used at
The map and the log	Concept	Map
Generated steps	Concept	Design

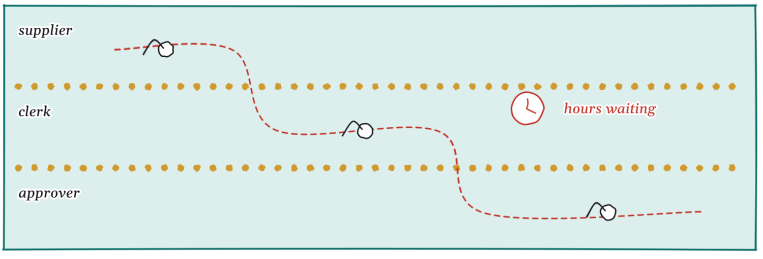


Like a pencil rubbing over a coin: *a process map means something only when it is pressed directly from the real work, with the people doing it.*

Process map

The process map records how a loop's work runs today. It is drawn as swimlanes: one row per role, team or outside party, with steps running left to right in the order they happen. Each step records the system or channel it uses, the minutes of hands-on work it takes, and the hours the case waits before the next step begins. Workarounds and rework are marked.

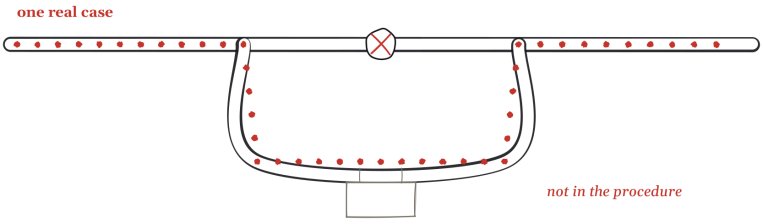
The map is drawn with the people who do the work, not with those who manage it. Managers describe the process as designed. Operators describe it as it runs. For one loop, eight to twenty steps is usually the right grain.



Like a swimming pool's lanes: *each role swims its own lane, the case moves left to right, and every crossing and pause is marked.*

Walk-through

A walk-through follows real cases through the map, from arrival to outcome, including at least one that went wrong. It is the quickest way to find steps the official process omits. At the manufacturer, the walk-through revealed that half the invoices still arrived as PDFs that a clerk printed and scanned, and that budget owners approved on paper every Friday. Neither appeared in the procedure.

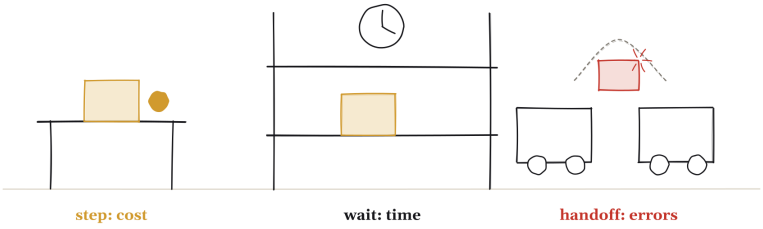


Like dye dropped into a plumbing system: *follow one real case through the pipes and it shows the bypass no drawing ever recorded.*

Step, wait and handoff

A step is a piece of work someone does. A wait is the time a case spends between steps, untouched. A handoff is a point where the case moves from one lane to another.

These three are the grammar of a process map. Steps are where cost sits. Waits are where elapsed time sits, usually most of it. Handoffs are where errors and delays gather, because each one requires the receiver to understand what the sender meant.

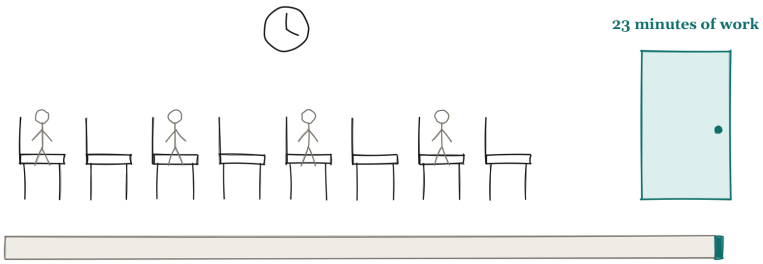


Like a parcel's journey: *a few minutes on the sorting bench, days on a shelf, and a knock each time it changes vans.*

Flow efficiency

Flow efficiency is the hands-on time a case receives divided by the total elapsed time from arrival to outcome. In most office processes it is very low, often under 5%. At the manufacturer, twenty-three minutes of work took sixteen days, a flow efficiency well under 1%.

The measure matters because it shows where improvement is possible. An executor that does each step faster leaves the waits untouched. A loop with 1% flow efficiency can only become much faster by removing waits, which means removing handoffs, batch steps and queues.

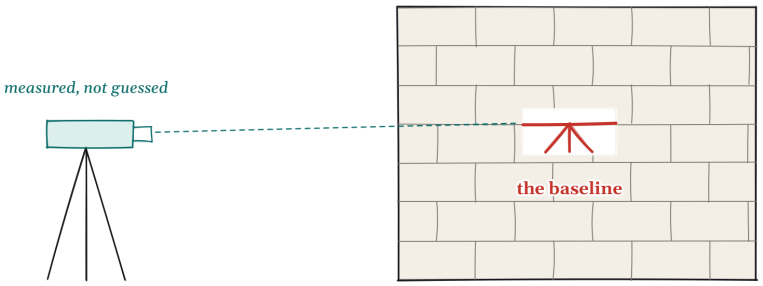


Like a clinic visit: *twenty-three minutes with the doctor after sixteen days in the waiting room, so a quicker doctor barely changes the total.*

Baseline

The baseline is the measured performance of a loop before it changes. It records five figures on a real sample of cases: volume, fully loaded cost per case, elapsed time, the share of cases raising an exception, and the share right the first time. It is signed by the process owner at Step 2.

The baseline is the reference for every later claim. It must be measured, not estimated. The usual period is several weeks, and the sample should include the tail.



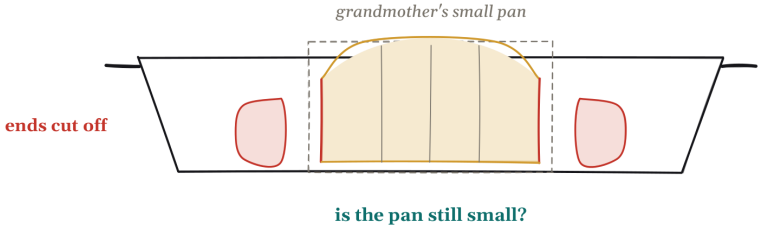
Like a surveyor's benchmark carved into a wall: *every later height is measured from it, so it has to be measured carefully, not guessed.*

Compensation

A compensation is a step that exists only to work around a constraint. A batch step exists because a system once ran overnight. A handoff exists because two teams once sat apart. A second review exists because errors were once frequent. A printed form exists because approvals were once on paper.

Compensations accumulate over years and become indistinguishable from real work. People are trained in them, controls rely on them, and improvement projects polish them. Attach an executor to a process full of compensations and it will run the workarounds faster, while every wait between them stays.

For each step, ask which constraint it compensates for and whether that constraint still applies. If it is gone, strike the step. If it still applies, keep the step and record the constraint, because a junction or the standard may meet it better. The register of these answers is signed at Step 3.

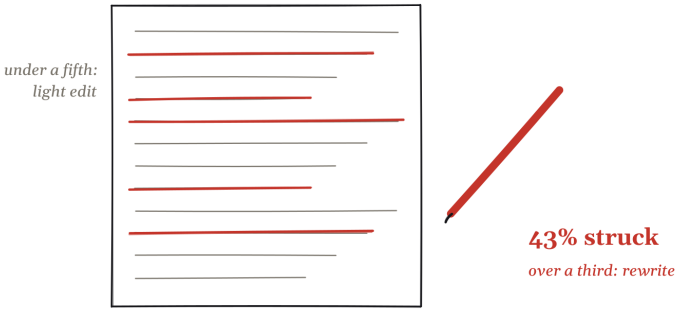


Like cutting the ends off a roast because grandmother's pan was small: *for each step, ask whether its constraint still applies, and strike it if not.*

Strip ratio

The strip ratio is the share of a map's steps struck as compensations. Under about a fifth, the process is mostly real structure, and embedding is largely a

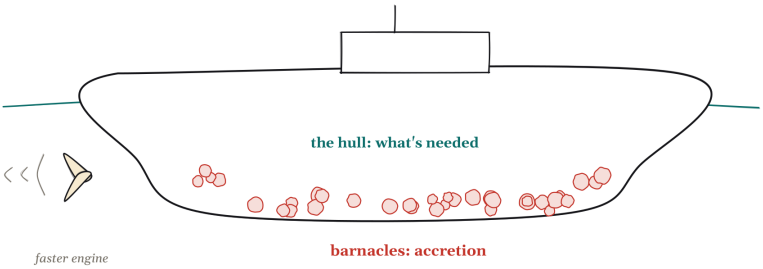
matter of letting an executor run what remains. Over about a third, the process is mostly accumulated workaround, and embedding is a redesign. The manufacturer's invoice loop had a strip ratio of 43%.



Like a manuscript marked with a red pen: *the share of lines struck tells you whether it needs a light edit or a rewrite.*

Topology and accretion

The steps an outcome genuinely requires are the loop's topology. The layer of compensations around them is accretion. The distinction gives the strip ratio its meaning, and it carries a warning. Automating a topology produces a faster loop. Automating an accretion produces a faster version of a loop that should not exist in its current form.

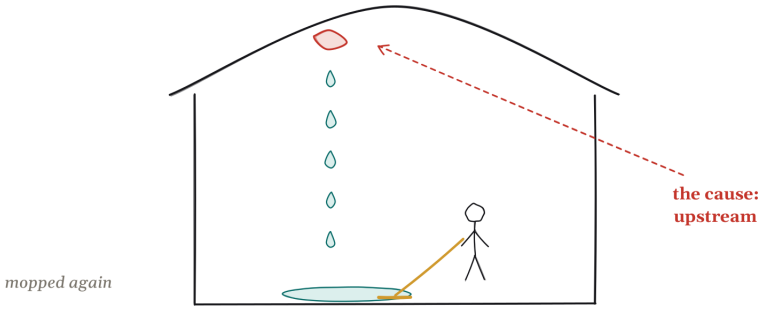


Like a ship's hull crusted with barnacles: *the hull is what the voyage needs, the barnacles are accretion, and a bigger engine just drags them faster.*

Rework

Rework is any step that redoes something that should have been right the first time: chasing a missing document, correcting a coding error, re-requesting information already supplied. It is marked on the map separately from compensations. A compensation is a step designed around a constraint.

Rework is a step caused by an error upstream. Rework usually points to a problem in a neighbouring loop or in the context the loop receives.

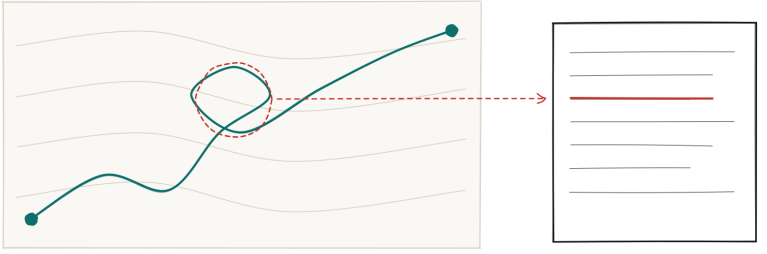


Like mopping the same floor again and again under a leaking roof: *rework is the mop, and the cause usually sits upstream in another loop.*

The map and the log

A process map records what people say happens. Most company systems also record what actually happened, as event logs: time-stamped records of each step each case went through. Process mining is the analysis of those logs to reconstruct the real flow of work. It was developed by Wil van der Aalst and others from the late 1990s.

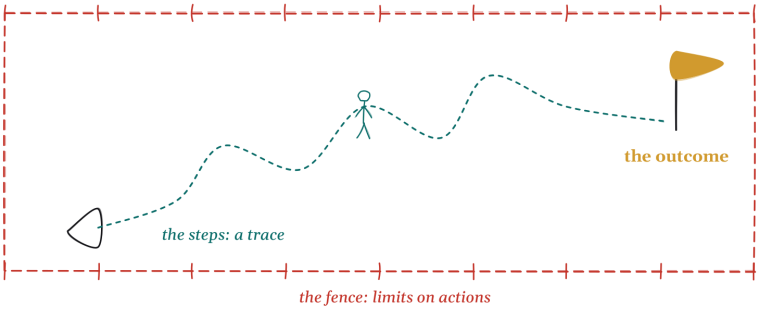
The two views complement each other. The log shows variants, loops and waits that nobody mentions in a workshop. The map explains why they happen, and marks which steps are compensations. Where logs exist, the map should be checked against them. Where they disagree, the log is usually right about what happened, and the people are usually right about why.



Like a hiker's GPS track beside their trail diary: *the track shows where they actually went, the diary explains why they turned back.*

Generated steps

In some loops the executor plans its own steps for each case: researching a proposal, investigating an incident, drafting an analysis. There is no fixed sequence to map in advance. In these loops the design moves up a level. The outcome, the rules, the limits on actions, the junctions and the standard become the design, and the steps become a trace of what the executor did. Mapping the current process still matters, because it gives the baseline and shows what the work achieves today.



Like an orienteering race: *you set the finish, the boundary and the rules rather than the route, and the route is recorded afterwards as a trace.*

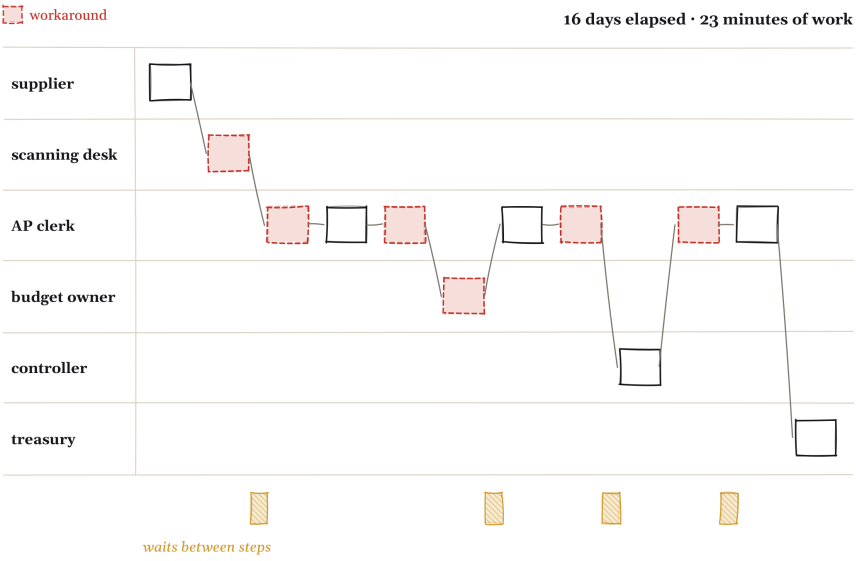


FIGURE 19 Work and process. A loop as it really runs: lanes, steps, workarounds in red and waits between steps. Most of the elapsed time is waiting.

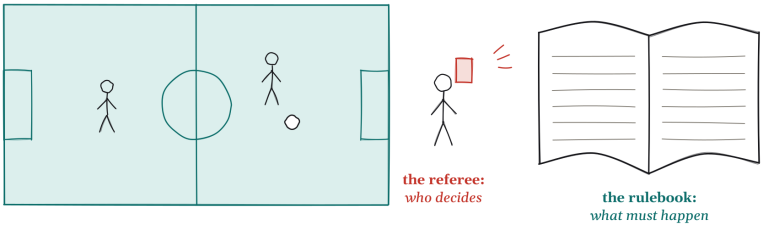
PART FOUR

Designing the Loop

XX

Rules and Decisions

This area covers what a loop must follow and who decides what inside it. It is where the company's unwritten agreements are written down, and where most of the important findings of an embedding programme appear.

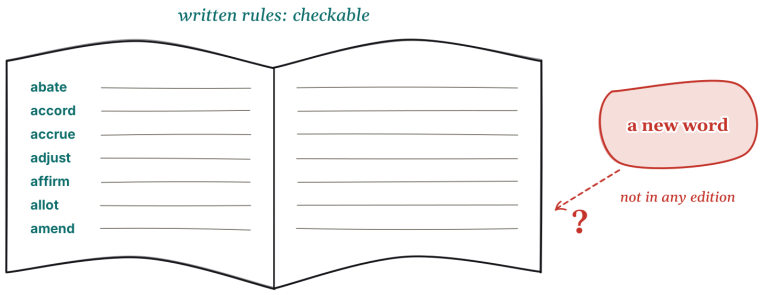


Like a football match: *the rulebook says what every player must follow, the referee says who decides, and writing both down shows where nobody agreed.*

Where it comes from

This area builds the symbolic half of the neurosymbolic loop. Its foundations are the laws of thought and their limits: rules written explicitly can be checked and audited, and no rule set is complete. The four classes of decision say where rules suffice, where judgement within limits suffices, where people must decide, and where the organisation itself must first decide.

Concept	Kind	First used at
Rule and rule source	Concept	Sort
Rule register	Instrument	Sort
Contradiction	Concept	Sort
Decision	Concept	Sort
The four classes of decision	Concept	Sort
Decision register	Instrument	Sort
Contested decision	Concept	Sort
Contested-decision log	Instrument	Sort
Decision rights	Concept	Sort, Own
Separating decisions from process	Concept	Sort, Design

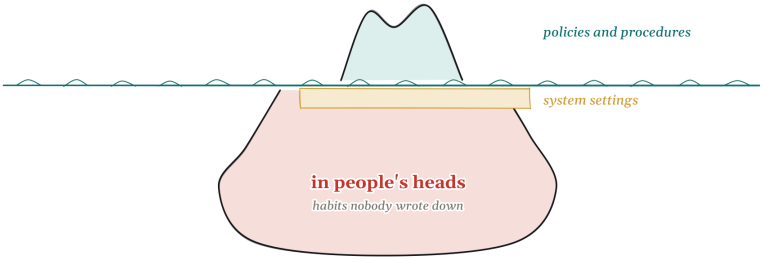


Like a dictionary: *every word written in it can be looked up and checked, yet new words keep arriving that no edition contains.*

Rule and rule source

A rule is a statement of what must or must not happen in a case: an approval limit, a tolerance, an eligibility condition, a required check. Rules live in three places. Some are documented in policies and procedures. Some are configured in systems, as tolerances, validations and workflows. And some exist only in the heads of experienced staff, as habits nobody wrote down.

The third source is the one most often missed and the one that matters most to an executor. A clerk who knows that one supplier always invoices freight separately is applying a rule. If nobody writes it down, the executor cannot apply it.

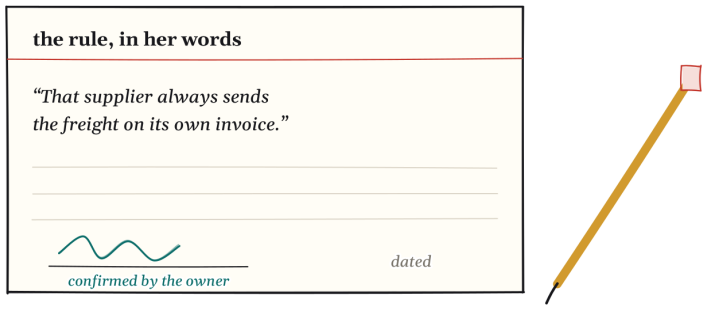


Like an iceberg: *a few rules show above water in policies, some sit at the waterline in system settings, and most hide below in people's heads.*

Rule register

The rule register collects every rule of a loop in one place, with its source, its owner and the date it was confirmed. Rules from people's heads are written in the words of the person who holds them, then confirmed by the owner. The

register is signed at Step 4 and becomes the main source of the executor's instructions.

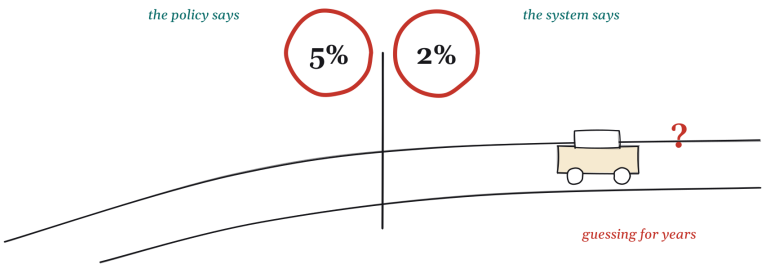


Like writing down a family recipe: *the cook's habit goes on the card in her own words, then it is checked, signed and dated so anyone can follow it.*

Contradiction

A contradiction is a pair of rules that cannot both be followed: the policy says one thing, the system enforces another, or two experienced staff apply different rules to the same case. At the manufacturer, the policy allowed a 5% price difference before approval was needed, while the ERP blocked anything over 2%. Clerks had been releasing the difference by hand for years.

Contradictions are findings, not noise. Each one is a place where the current process has been guessing, and where an executor would have to guess. Each needs an owner to decide which rule stands.

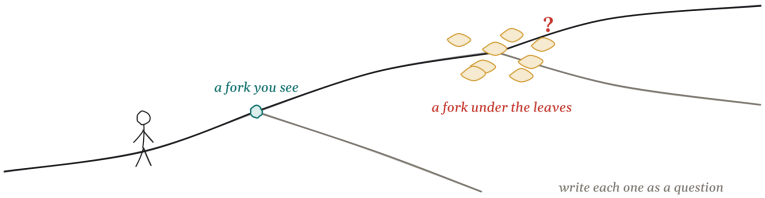


Like two speed signs on the same road that disagree: *drivers have been guessing for years, and an owner must decide which sign stands.*

Decision

A decision is any point in a loop where the work could go more than one way. Is this a duplicate? Does it match within tolerance? Which cost centre? Approve or hold? Decisions are best written as questions. Listing them is

harder than it seems, because in a process built for people, many decisions are buried inside steps and taken without anyone noticing.



Like a woodland path with forks: *some forks are plain to see, others lie under the leaves, and walkers turn there without noticing a choice.*

The four classes of decision

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

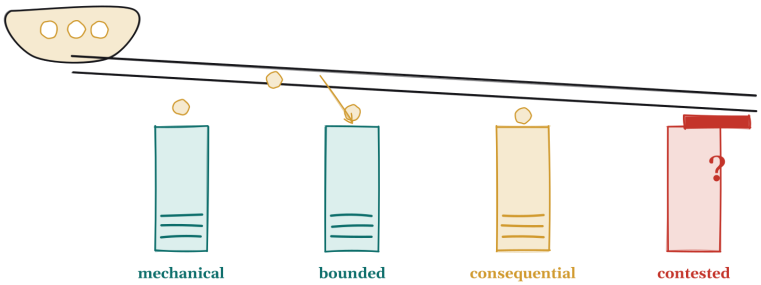
A mechanical decision follows a written rule completely. The executor can take it alone.

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

A consequential decision carries a high cost of error, or leads to an action that cannot be undone. A person always decides, with the executor preparing the evidence.

A contested decision is one the organisation has never agreed on. It cannot be designed; it must be settled.

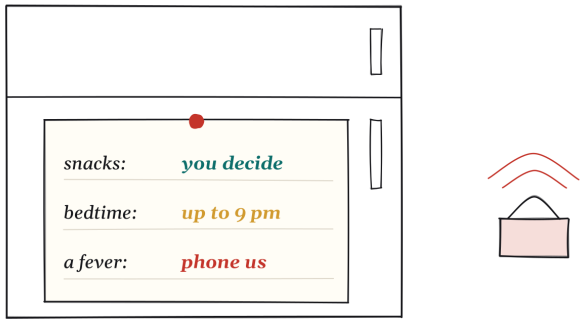
The class depends on the state of the rules as well as the stakes. A decision is mechanical only if its rule is written and uncontradicted, and bounded only if someone has signed the limit. A common misreading is to class decisions by comfort: everything becomes consequential because nobody wants to let go, or everything becomes mechanical because a pilot looked good.



Like a coin sorter with four tubes: *each decision drops by its kind, and the fourth tube is capped until the company settles what the coin is worth.*

Decision register

The decision register lists every decision in a loop, as a question, with its class, the reason for the class, and the limit if it is bounded. It is signed at Step 4. It is the single most useful document for anyone trying to understand what an executor in the loop is allowed to do.



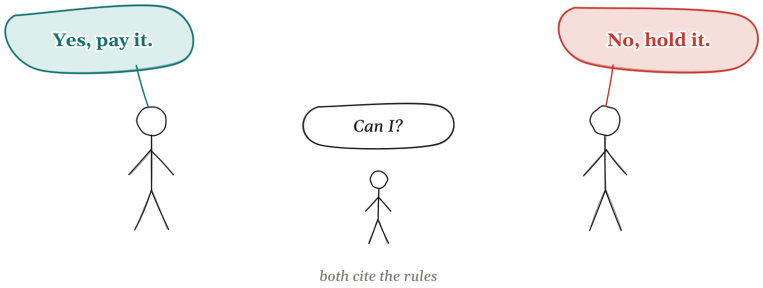
Like the note left for a babysitter: *what they may decide alone, what they may decide up to a limit, and when they must phone the parents.*

Contested decision

A contested decision is a decision the organisation has never actually agreed on. Two senior people would answer it differently, and both would cite policy. It has usually been handled for years by escalation, by habit, or by not looking. Pay a disputed invoice to protect supply? Grant a goodwill refund outside policy? Accept a late claim from a loyal customer?

No design can resolve a contested decision. An executor cannot be given a rule the company does not have. Someone with authority must decide, write it down and sign it. Contested decisions are found by asking, in the room, which

decisions two senior people would answer differently. They are the most important findings of Step 4, because each one would otherwise surface as an incident the first time an executor met it.

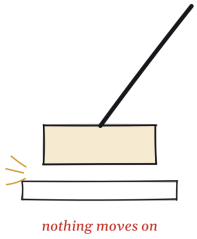


Like a child asking two parents the same question and hearing yes and no: *there is no rule to give the babysitter until the parents settle it.*

Contested-decision log

The contested-decision log records each contested decision: the question, the positions held, who settled it, the answer and the date. The lifecycle does not pass from Sort to Design while any entry is open. At the manufacturer, the log had one entry, on disputed invoices from single-source suppliers. The chief financial officer and the chief operating officer settled it in a thirty-minute meeting.

question	positions	ruling	date
pay a disputed invoice?	*pay / hold*	SETTLED	*signed*
a late claim?	*accept / refuse*	OPEN	



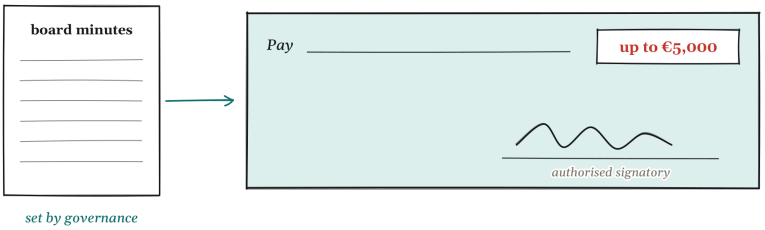
Like a court docket: *each dispute is entered with both positions, the ruling, who gave it and the date, and the list cannot move on while one stays open.*

Decision rights

Decision rights say who may take which decision, up to which limit. In most companies they are partly written, in approval matrices and delegations of authority, and partly assumed. Embedding forces them into the open. Each bounded decision needs a signed limit, which is a decision right delegated to

the executor. Each consequential decision needs a named person, who holds the decision right at a junction.

A common misreading is to treat decision rights as a matter for the executor's configuration. They are a matter for the company's governance. The configuration should reflect what the company has decided, not substitute for it.



Like signing limits on a company chequebook: *the board decides who may sign and up to what amount, and the printer only prints what was decided.*

Separating decisions from process

In 2015, the Object Management Group published DMN, a standard for modelling decisions separately from the processes that use them. Decision tables were written down once and called from many places. The idea was sound, but adoption stayed in pockets, partly because no executor could act on a decision table without a person translating it into steps.

The executor that decides changes that. Rules and decisions written down and sorted can be given to it directly. Separating decisions from process, the old idea behind DMN, becomes the practical core of Step 4.



Like a price list pinned on the shop wall: *written once and read by every till, instead of each cashier carrying the prices in their head.*

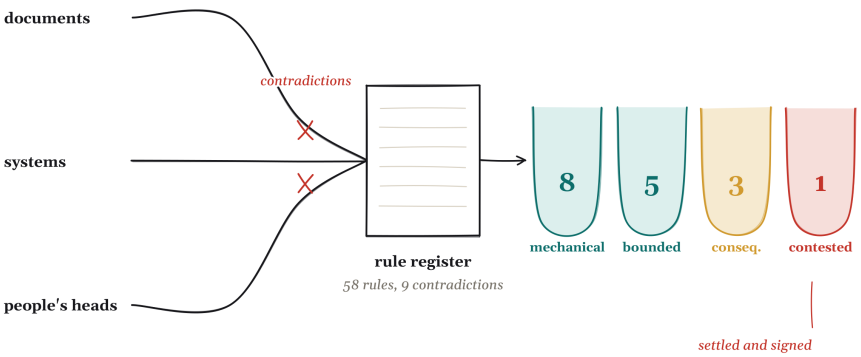
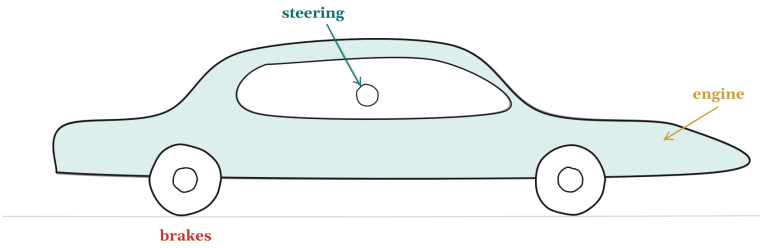


FIGURE 20 Rules and decisions. Rules from documents, systems and people's heads meet in one register, where contradictions surface; every decision is sorted into one of four classes.

XXI

The Executor

This area covers what does the work in an embedded loop: what an executor that decides is, what it is made of, what it can read and change, and how the business keeps control of each part. It is written for business readers. It explains each part well enough to make the decisions that part requires, and no further.



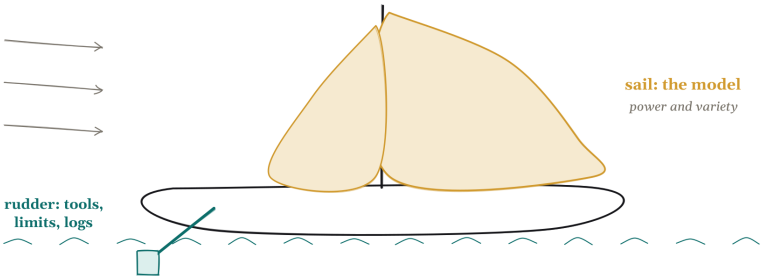
Like a car explained to a new owner: *enough about engine, steering and brakes to make each decision about driving it, and no further.*

Where it comes from

This area covers the neural half of the loop and the machinery that connects it to the symbolic half. A language model brings the variety rules lack. Tools, permissions, logs and the platform turn its proposals into accountable actions.

Concept	Kind	First used at
Executor	Concept	Read
Instruction-follower and decider	Concept	Read
Model	Concept	Design
Instructions	Concept	Design
Context and keepers	Concept	Design
Actions and reversibility	Concept	Design, Bound
Permissions	Concept	Bound
Platform	Concept	Design, Bound
The jagged frontier	Concept	Shadow
Pilot	Concept	Shadow

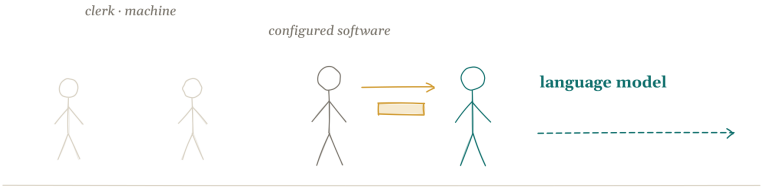
Concept	Kind	First used at
Build, buy or rent	Concept	Design



Like a sailing boat: *the wind in the sail brings the power and the variety, and the rudder, sheets and log turn it into a course someone answers for.*

Executor

The executor is whatever does the work in a loop. For most of the history of business it was a person, a machine following a sequence, or software applying configured rules. In an embedded loop it is, for much of the work, a language model with access to the company's systems, acting within limits the business has set.

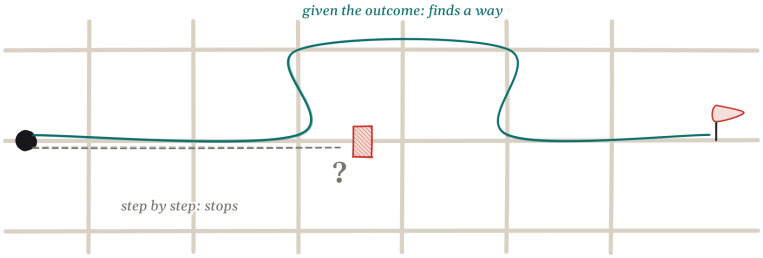


Like a relay race: *the baton of the work passed from people to machines to configured software, and is now handed to a language model.*

Instruction-follower and decider

An instruction-follower needs the work specified step by step in advance, and hands to a person wherever the specification runs out. A decider is given an outcome, rules and limits, and works out the steps itself. It can read a document it has never seen, apply a rule written in plain language, and reach a decision.

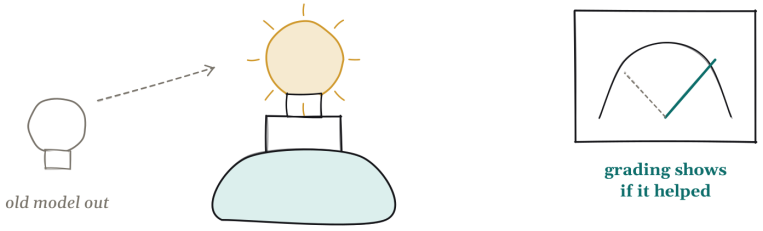
The difference changes how work is designed. For a decider, the important things to specify are no longer the steps. They are what counts as done, what it may not do, where it must stop and ask, and how its results will be graded.



Like a satnav versus a local driver: *the satnav stalls when its route is blocked, while the driver given the address and no-go streets finds a way.*

Model

The model is the language model at the centre of the executor, trained on large amounts of text to read, reason and write. Companies usually buy access to a model from a provider, or run an openly available model on their own infrastructure. Models improve quickly and are replaced often. A loop should be designed so that its model can be changed, and its grading should show whether the change helped.



Like a bulb in a standard socket: *the lamp is built so the bulb can be swapped, and a light meter shows whether the new one is better.*

Instructions

The instructions tell the model its job: the outcome, the rules, the limits, the junctions and what to do with exceptions. In a well-designed loop they come straight from the signed registers. A loop whose rules were never gathered can only give its executor vague instructions, and vague instructions produce inconsistent decisions.

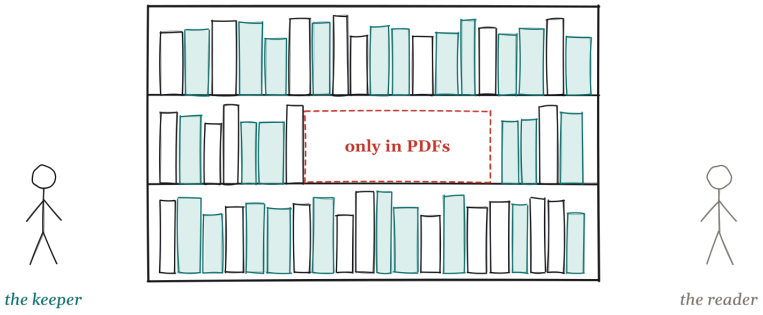


Like sheet music: *a signed score gets the same tune every time, while a note saying play something nice gets a different tune each night.*

Context and keepers

Context is the information the executor must read to act well on a case: the order, the receipt, the contract, the customer's history. It lives in the company's systems, and the executor reaches it through connections someone must build and maintain. Each source has a keeper, the person responsible for keeping it accurate.

A gap in context is a decision made blind. At the manufacturer, contract prices for about a third of suppliers existed only in PDF agreements. Procurement loaded the prices for the top two hundred suppliers into the ERP before shadow running.

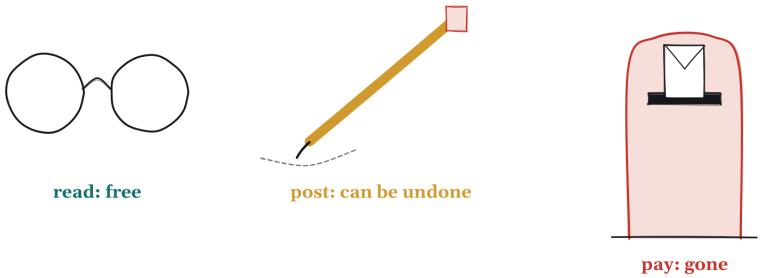


Like a library with a keeper for each shelf: *the reader can only act well on books that are there, and a missing shelf means deciding blind.*

Actions and reversibility

Actions are the changes the executor may make: posting a record, placing a hold, sending a message, releasing a payment. Each is described by the system it touches and one question: can it be undone, and at what cost? Reversibility

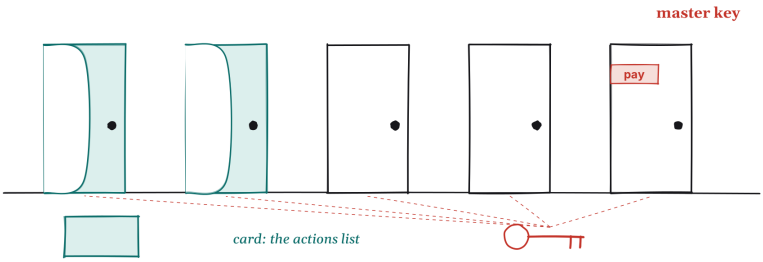
is the main input to where junctions go. Reading is cheap. Posting can be reversed. Releasing a payment usually cannot.



Like reading, writing in pencil and posting a letter: *reading costs nothing, pencil can be rubbed out, and a letter in the postbox cannot be called back.*

Permissions

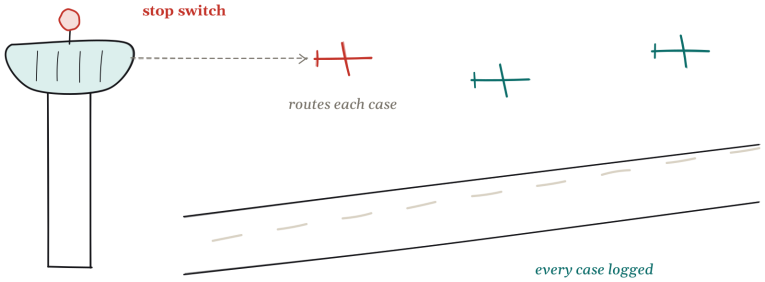
Permissions are the rights an executor holds in each system. They should match its list of actions exactly. Permissions granted broadly for a pilot "to get things working" are the commonest source of executors doing things nobody intended. Cutting permissions back to the actions list is part of Step 7.



Like a hotel key card coded for the rooms on its list: *a master key handed over just to get things working opens every door, payments included.*

Platform

The platform is the layer that runs the loop around the model. It routes each case, applies limits, sends consequential cases to junctions and exceptions to the queue, records every case, and holds the switch that stops the loop. Whether it is built or bought, the business should require three things of it: a complete log of every case, a switch the named person can use, and access for the grader to every result.

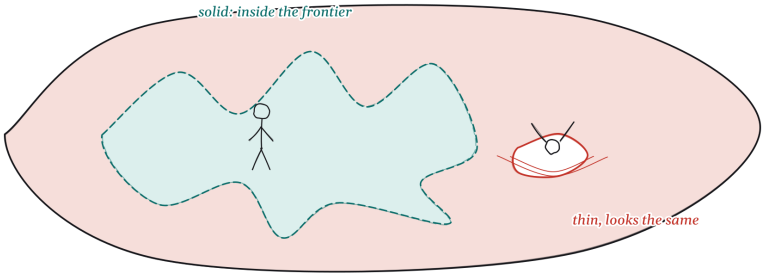


Like an airport control tower: *it routes every plane, sends the unusual ones to a person, logs every movement, and can stop the runway.*

The jagged frontier

Executors are not uniformly capable. A 2023 study of consultants using AI, by Fabrizio Dell'Acqua and colleagues, described a "jagged technological frontier". On tasks inside the frontier, AI help raised quality substantially. On tasks just outside it, people using AI did worse than those without it, because they trusted answers they should have checked. The frontier is uneven, and it is not visible in advance.

That is why shadow running reads agreement case by case and band by band, rather than trusting a demonstration. It is also why the tail stays with people until the evidence says otherwise.

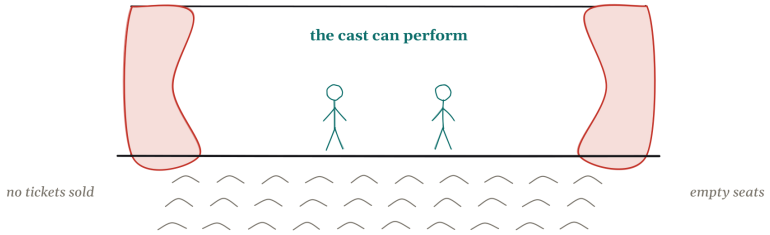


Like a frozen lake with uneven ice: *solid in places, thin just beyond, and the edge cannot be seen, so you test each patch before you trust it.*

Pilot

A pilot is a test of whether an executor can do a task, on a prepared set of cases. It usually builds the model and instructions well, connects some context, and leaves out real actions, the platform's switch and log, and the grader. A pilot is valuable evidence and a small part of the work. In the

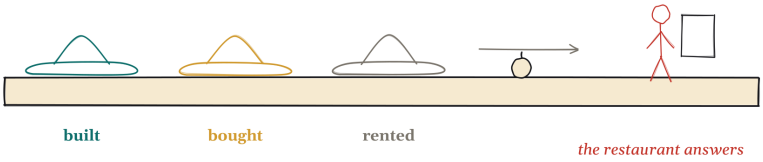
lifecycle it belongs inside Step 6, as the executor in shadow, after Steps 1 to 5 have been done.



Like a dress rehearsal: *real proof the cast can perform, but with empty seats, no tickets sold and no reviewers, so the real run is still ahead.*

Build, buy or rent

Every part of the executor can be built, bought or rented, and the choice changes over time. The choice does not change the business's responsibilities. Whoever supplies the executor, the company must sign its rules, sort its decisions, place its junctions, write its standard and measure its economics. A vendor-run loop needs contract terms that give the company the log, the switch, access for its grader, and notice before the model changes.



Like a restaurant's kitchen pass: *dishes cooked in-house, bought in or catered all cross the same pass, and the restaurant answers for every plate.*

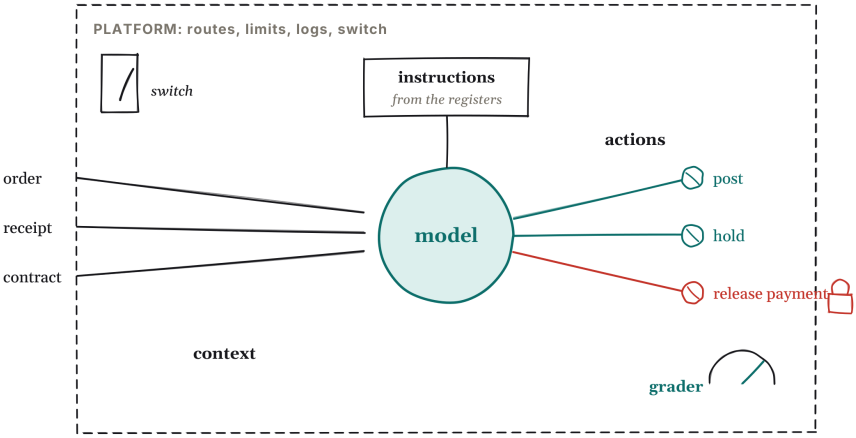
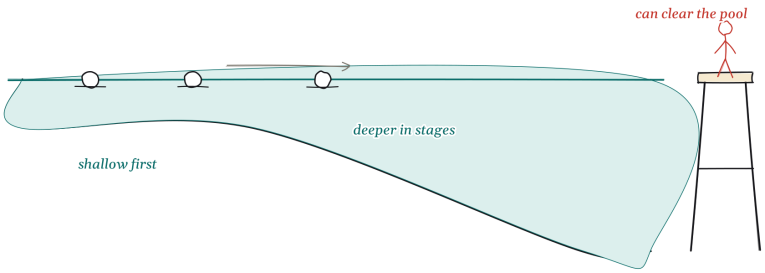


FIGURE 21 The executor. A model briefed by instructions from the registers, reading context, acting through permissioned tools, inside a platform that logs and can be switched off, with a grader on every outcome.

XXII

Control and Acceptance

This area covers how a company knows an embedded loop is doing good work, where people decide, how trust is earned in stages, and how the loop is stopped when it should be. It holds the concepts that make an executor that decides safe to rely on.



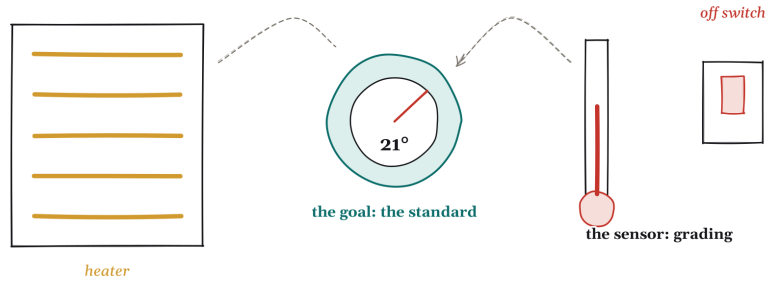
Like a lifeguard at a pool: *watching that the swimming goes well, standing by the deep end, letting swimmers in deeper by stages, and able to clear the water.*

Where it comes from

This area is cybernetics applied. The standard is the goal, grading is the sensor, and junctions add the variety of human judgement at the tail. The switch-off is the means of opening the loop safely. Without this area, an embedded loop runs open: acting without knowing whether its actions are right.

Concept	Kind	First used at
Junction	Concept, instrument	Design
Switch-off	Instrument	Design, Bound, Core
Exception queue	Instrument	Design
Standard	Instrument	Design
Grading	Concept	Design, Shadow
Shadow running	Concept	Shadow
Agreement, body and tail	Measure	Shadow
Bounded scope	Concept	Bound
Scale Gates	Concept	Design to Core

Concept	Kind	First used at
Drift	Concept	Core
Override rate	Measure	Core
Controls that survive and controls that invert	Concept	Design, Bound
The ironies of automation	Concept	Own, Core



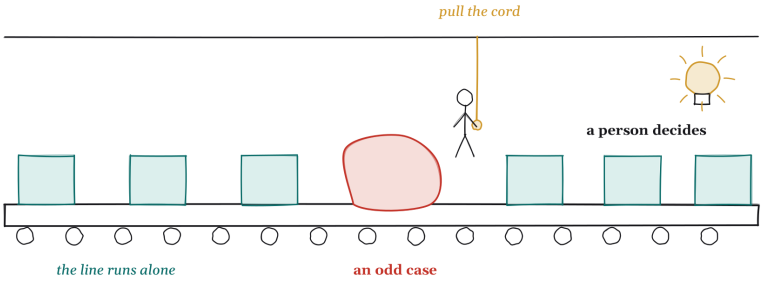
Like a thermostat: *the setting is the standard, the thermometer is the grading, a hand can override it, and the switch opens the loop safely.*

Junction

A junction is a point where a person decides instead of the executor. It is defined by three things: who decides, what evidence the executor puts in front of them, and how a wrong decision is reversed. Junctions are placed by the reversibility and cost of the action, not by how comfortable anyone feels. Every consequential decision and every irreversible action needs one.

The ancestor of the junction is the cord on Toyota′s assembly lines that any worker could pull to stop the line when something looked wrong. The routine runs on its own, an abnormal case stops it, and a person decides.

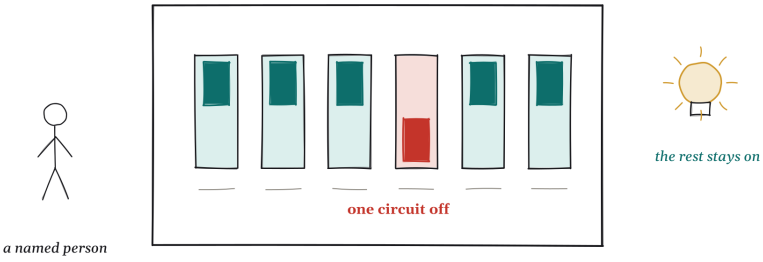
A common misreading is to treat any human approval as a junction. An approval with no designed evidence, no clear holder and no way to reverse the decision is a rubber stamp.



Like the cord on Toyota's assembly line: *the line runs on its own until an odd case appears, a worker pulls the cord, and a person decides.*

Switch-off

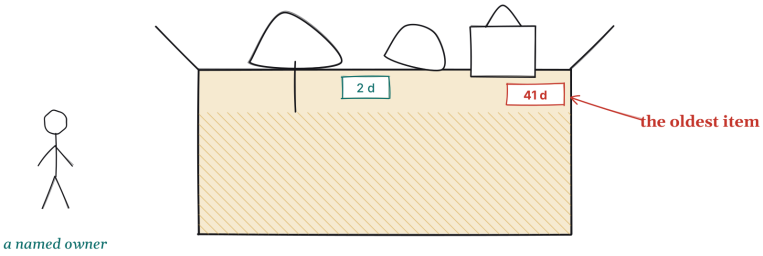
The switch-off is the means by which a named person stops the loop, and returns its cases to people, within a stated time and without losing any. It is designed at Step 5, tested before the first live case at Step 7, and tested on a schedule in Core. Using it on part of the scope for a few days, when drift or an incident calls for it, is normal operation, not failure.



Like a fuse box with labelled switches: *a named person turns one circuit off for a few days, the rest of the house stays lit, and nothing is lost.*

Exception queue

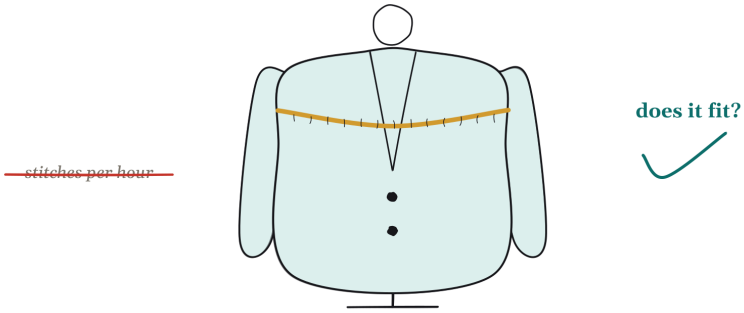
The exception queue is the single place where cases outside every rule go. It has a named owner, a time limit, and a weekly review that turns repeated exceptions into new rules. The age of its oldest case is one of the most revealing measures of a loop's health.



Like a station's lost-and-found box: *one place for things that fit nowhere, one owner, a weekly sort, and the oldest tag says how healthy it is.*

Standard

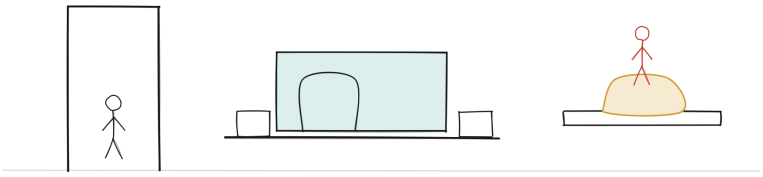
The standard is the written definition of an accepted outcome, with the method of grading it. Its criteria describe the outcome, not the activity. "Right supplier, amount, tax and cost centre, no duplicate, ready within terms" is a standard. "Invoices processed per hour" is not. Customer-facing loops include the customer's view in the criteria, through complaints, repeat contacts and reopened cases.



Like a tailor's fitting: *the suit is judged by whether it fits the customer, not by how many stitches were sewn per hour.*

Grading

Grading applies the standard to outcomes. The usual pattern combines three layers. Rules check every case for what can be checked mechanically. A second model grades every case against the criteria. And a small human audit, weighted towards the tail, calibrates the model grader. Grading every outcome, not a sample, is what makes it possible to run an executor that decides, because there is no procedure to audit instead.



Like airport security: *an arch checks every passenger, a scanner checks every bag, and an officer opens a few by hand to keep the scanner honest.*

Shadow running

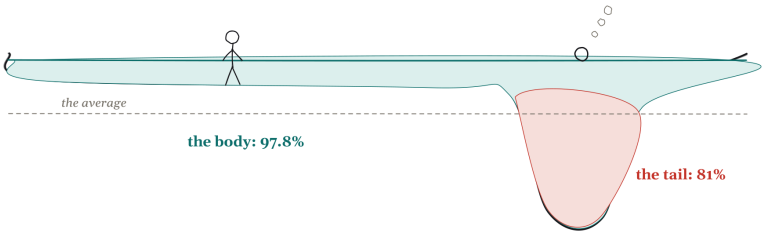
Shadow running puts the executor beside the current process on real cases, reaching its own results without acting on anything, so they can be compared with the people's results. It runs long enough for the tail to arrive in useful numbers, typically four to eight weeks. Every disagreement is classified. The executor may have been wrong, or the person may have been wrong. Both may be defensible, or the standard may be unclear.



Like a new clock hung beside the old one for weeks: *it keeps time, nobody sets the trains by it, and every difference is written down.*

Agreement, body and tail

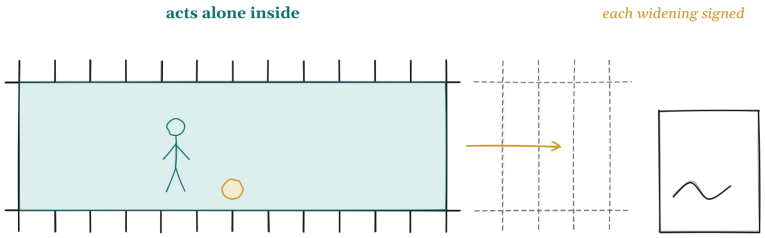
Agreement is the share of cases where the executor's result matches the accepted result. It is always reported separately for the body and the tail, because a single overall figure hides the difference that matters. At the manufacturer, agreement was 97.8% on invoices under €5,000 and 81% on invoices over €50,000.



Like a river whose average depth is one metre: *the wide shallows hide the deep channel, so you read the shallows and the channel separately.*

Bounded scope

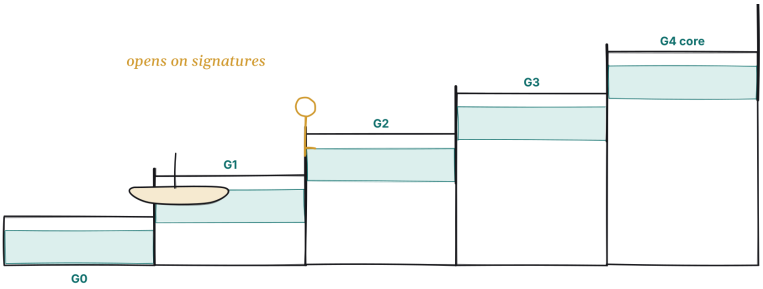
Bounded scope is the written set of conditions under which the executor may act alone: values, tolerances, counterparties, case types. It is drawn from shadow evidence and signed by the control owner at Step 7. Every widening is recorded and agreed. A common misreading is to set the scope by ambition, because the pilot handled large cases well in a demonstration.



Like a playground fence: *the child plays alone inside it, the fence follows what was seen to be safe, and moving it out is agreed and signed.*

Scale Gates

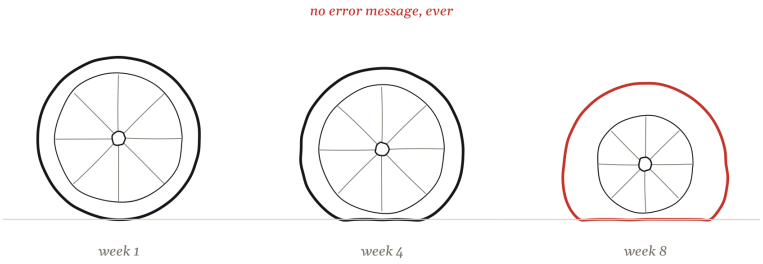
The Scale Gates are five stages a loop passes on its way to the core, each opened by particular signatures. At G0, Mapped, every node of the loop's canvas is drafted. At G1, Shadow, the outcome, compensations, rules, context and standard are signed. At G2, Bounded, the actions, decisions, junctions and exceptions are signed too. At G3, Owned, the people and economics are also signed. At G4, Core, all twelve are signed. Scale follows signatures, not confidence.



Like a flight of canal locks: *the boat rises one chamber at a time, and each gate opens when its keys are turned, not when the crew feels ready.*

Drift

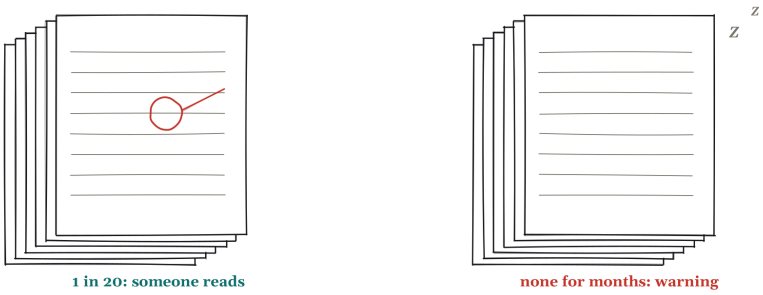
Drift is a gradual change in the quality of an executor's outcomes, caused by changes in its inputs, its model or the people around it. It produces no error messages. It shows as a slow fall in the share of outcomes meeting the standard, often in one segment first. It is caught by grading every outcome, by a weekly human sample that checks the grader, and by reading the results body and tail separately each month.



Like a slow puncture: *no bang and no warning light, just a tyre a little softer each week, caught only by checking the pressure regularly.*

Override rate

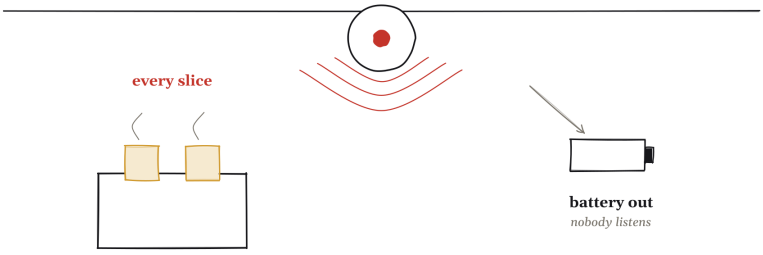
The override rate is how often junction holders overrule the evidence the executor puts in front of them. A rate near zero for months is a warning that the junction has become a rubber stamp. A rate that is above zero and explained is healthy. At the manufacturer, the controller overrode about one large invoice in twenty, usually because of information the executor could not see.



Like a proofreader: *one correction in twenty pages shows someone is reading, while months without a single mark are the warning.*

Controls that survive and controls that invert

Most controls in a process were designed for people, and they fare differently with an executor that decides. Some survive unchanged, such as the call-back on changed bank details. Some invert. Segregation of duties, which separates the person who records from the person who approves, becomes a separation between the executor and the standard it is graded against, with junctions at the consequential points. A four-eyes review becomes a grader plus a junction. Sampling becomes grading every outcome. And some controls become dangerous if left unchanged. "Review everything", applied to an executor's thousands of decisions a day, guarantees that reviewers stop reading.



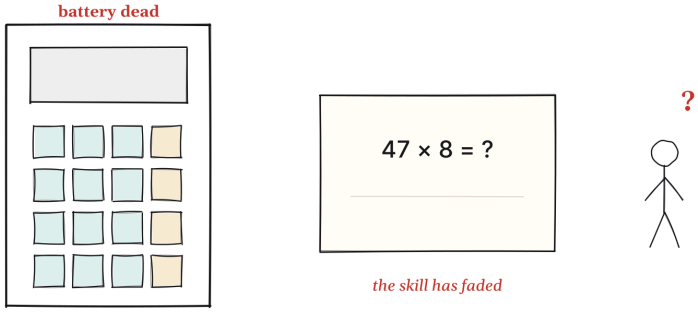
Like a smoke alarm that sounds at every slice of toast: *a control meant for rare events, applied to everything, teaches the house to take the battery out.*

The ironies of automation

In 1983, the engineer Lisanne Bainbridge described the "ironies of automation". The more a system is automated, the more crucial and the more difficult the human role becomes. People left to monitor automated systems lose the skills they need when the system fails, and they find it hard to stay alert to rare problems. Later research on automation bias found the same

pattern: people tend to accept a machine's recommendation even when the evidence suggests otherwise.

These findings shape three design choices: junction evidence that asks the holder to look before agreeing, override rates read as a health signal, and a deliberate plan for people to keep the skills the tail requires.



Like years of leaning on a calculator: *mental arithmetic quietly fades, and it is needed most on the one day the battery dies.*

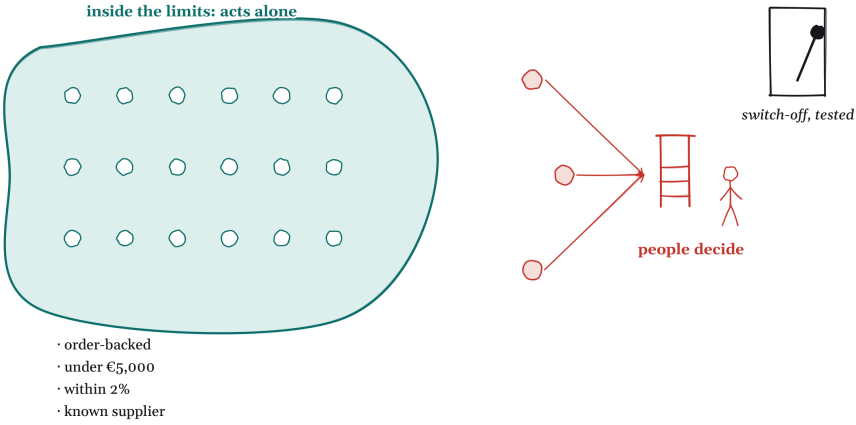


FIGURE 22 Control and acceptance. The executor acts alone only inside a signed scope; cases outside it pass through junctions to people, and the switch-off is tested before the first live case.

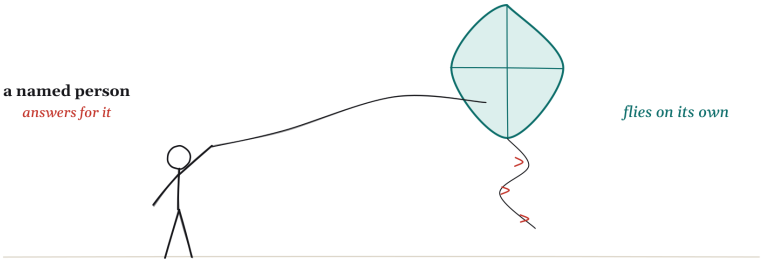
PART FIVE

Owning the Result

XXIII

People and Accountability

This area covers who answers for an embedded loop, who decides at its junctions, whose work changes, and how people keep and build the judgement the work still requires. An executor cannot be accountable. Everything it does is answered for by a named person, and this area says which one.



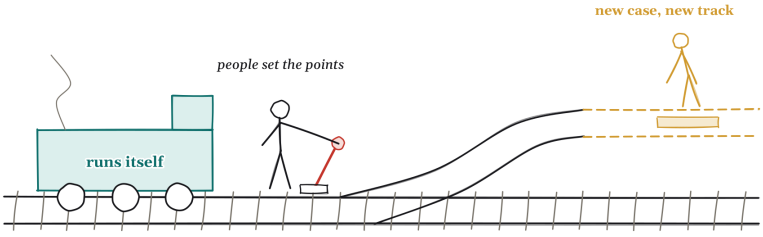
Like a kite on a string: *the kite flies on its own, but the person holding the string answers for where it goes.*

Where it comes from

This area keeps the strange loop honest. People design the symbolic layer, hold the junctions, extend the rules when the world shows them a new case, and answer for the results. An executor cannot be accountable; every decision it takes is answered for by a named person.

Concept	Kind	First used at
Executive sponsor	Role	Read
Process owner and loop owner	Role	Map, Design
Control owner	Role	Bound
Junction holder	Role	Design, Own
Exception owner	Role	Design
Grader	Role	Design, Shadow
Adoption owner	Role	Own
Business engineer	Role	Read to Scale
Signature	Concept	All steps

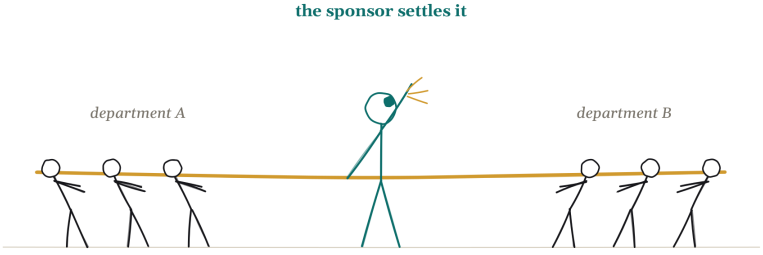
Concept	Kind	First used at
Loop charter	Instrument	Own
Role change plan	Instrument	Own, Scale
Learning the tail	Concept	Own
Loop roles and the RACI chart	Concept	Own



Like a railway: *the train runs itself, but people lay the track, set the points, and extend the line when a new town appears.*

Executive sponsor

The executive sponsor is the senior person who commissions embedding across a set of loops and can settle disagreements between departments. The sponsor signs the Work Loop Map, the contested decisions that need their authority, the map review and the role change plan. A sponsor who cannot settle a contested decision is not senior enough for the loops in question.

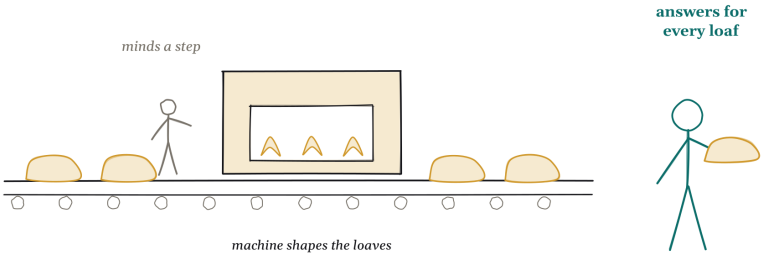


Like a tug-of-war referee: *two departments pull against each other, and only someone senior enough can blow the whistle and settle it.*

Process owner and loop owner

The process owner is responsible for how a process runs today. During Design, the process owner usually becomes the loop owner, who answers for the loop's outcome, including the parts the executor produces. The change of title marks

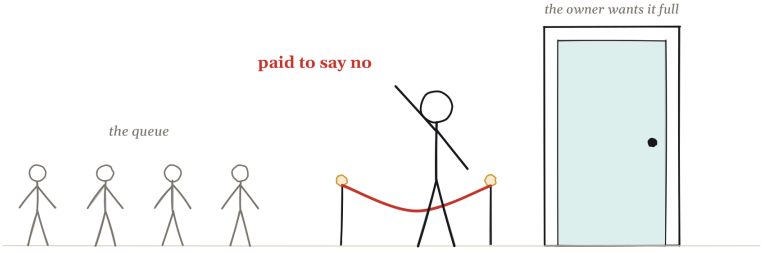
a change of responsibility. A process owner looks after steps. A loop owner answers for results.



Like a bakery line: *one person minds each step of the oven, but the head baker answers for every loaf that leaves, including those the machine shaped.*

Control owner

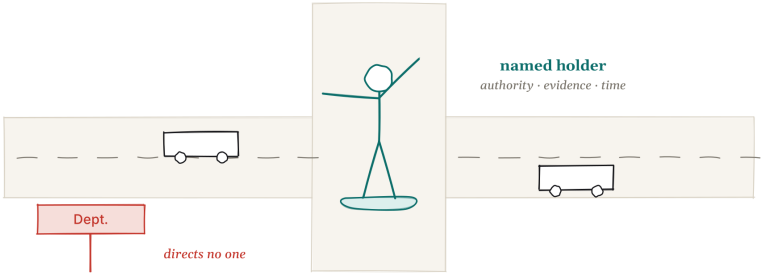
The control owner is responsible for the company's controls over a process, such as the financial controller or the head of risk for that area. The control owner signs the gate review at Bound and agrees each later widening of scope. The role is deliberately separate from the loop owner, who naturally wants the loop to run. The person who signs the limits should be the one whose job is to say no.



Like a club doorman: *the owner wants the room full, so the person who decides who gets in is the one whose job is to say no.*

Junction holder

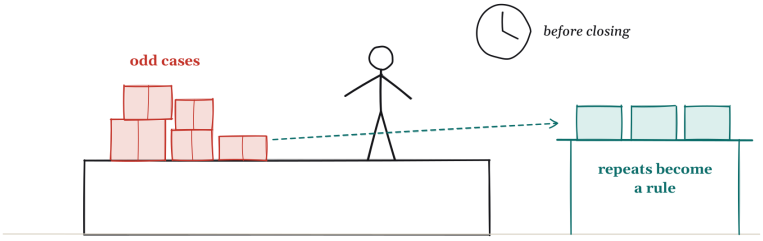
A junction holder is the named person who decides at a junction. Holders need the authority to decide, the evidence to decide well, and the time to do it. A junction whose holder is listed by department, not by name, has no holder.



Like a busy crossroads: *a named officer with authority directs the traffic, while a sign that only names a department directs no one.*

Exception owner

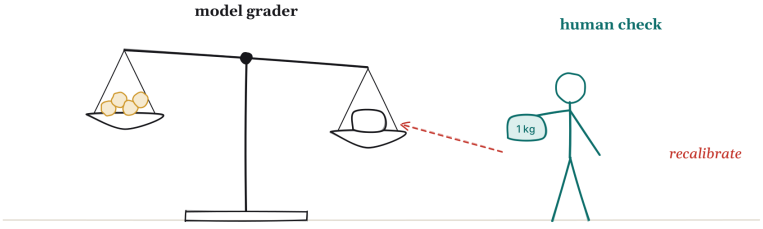
The exception owner works the exception queue, keeps it within its time limit, and runs the weekly review that turns repeated exceptions into rules.



Like a returns counter: *one clerk clears the odd parcels before closing time, and a parcel that keeps coming back gets its own shelf.*

Grader

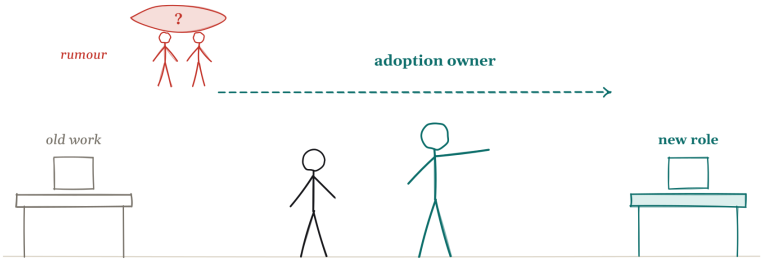
The grader is the person who keeps the standard and its grading calibrated: reviewing the human audit, comparing it with the model grader, and adjusting the criteria when the business changes. The grader signs the shadow report with the loop owner.



Like a weights inspector at a market stall: *she checks the stall's scale against her own reference weight and adjusts it when the two disagree.*

Adoption owner

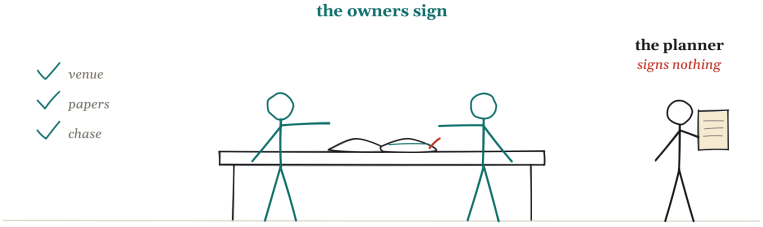
The adoption owner is responsible for the people whose work changes: agreeing the changes openly, helping people into new roles, and designing how new staff will learn the work. In most programmes this role is missing, and the change to people's work is left to rumour.



Like an office move: *someone has to show each person to their new desk, or the move happens by rumour in the corridor.*

Business engineer

The business engineer runs the lifecycle for a loop. They book the sessions, prepare the documents, chase the signatures and know what each gate requires. They sign nothing. Owners and control holders sign. The business engineer makes sure there is something worth signing. In companies running many loops, business engineers form a small function that owns the method, not the loops.

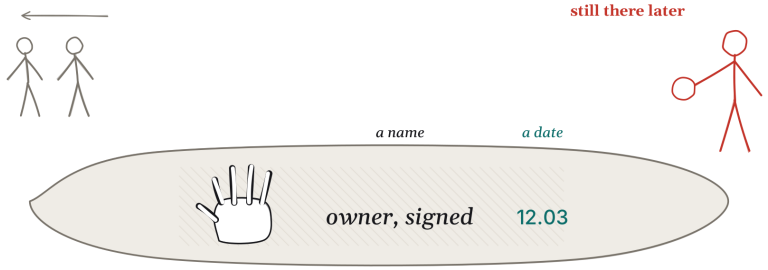


Like a wedding planner: *books the venue, prepares the papers and chases everyone, but never signs the register; the couple do.*

Signature

A signature is a named person's dated, recorded acceptance of a document as true or as decided. It does not need ink. It needs a name, a date and a record that survives the meeting. Signatures turn agreement into accountability.

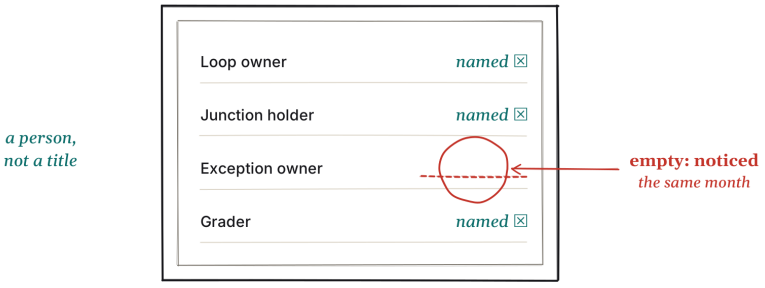
When an auditor, a regulator or a new manager asks why a loop behaves as it does, the signatures are the answer.



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

Loop charter

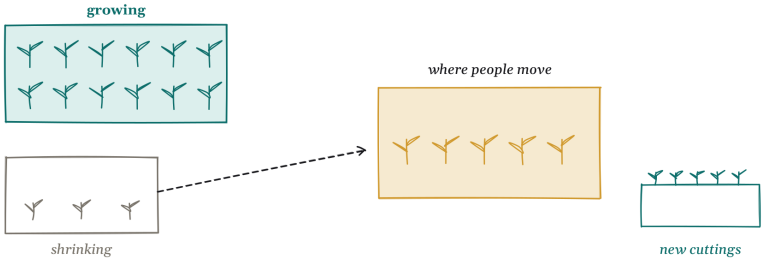
The loop charter names every role a loop needs with a person, not a title. It records the agreed changes to people's work and the learning path for new staff. It is signed by the loop owner at Step 8 and checked at every operating review, so that a role left empty by someone's move is noticed the same month.



Like a theatre cast list on the stage door: *every part carries an actor's name, so a blank slot is noticed before the next show.*

Role change plan

The role change plan is the map-wide view of how work is changing across all embedded loops. It records which roles are growing, which are shrinking, where people are moving, and how expertise will be rebuilt. It is signed by the sponsor at Step 10, because it commits the company, not just one team.

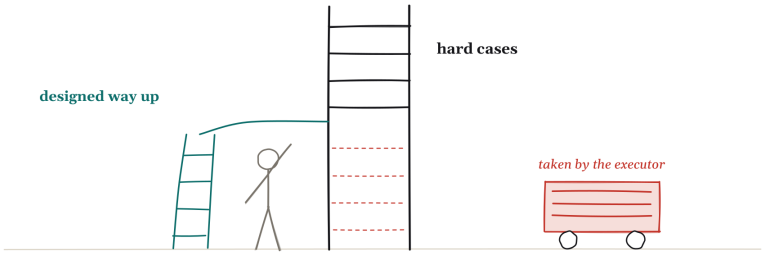


Like a replanting plan for a whole garden: *it shows which beds grow, which shrink, where plants move and where new cuttings are raised.*

Learning the tail

In most processes, people learned their judgement by handling simple cases until they were ready for hard ones. When an executor takes the simple cases, that path disappears. A company that does not replace it will, within a few years, have no one able to hold its junctions.

Research points both ways. A large study of customer-support agents by Erik Brynjolfsson, Danielle Li and Lindsey Raymond found that an AI assistant raised productivity by about 14% on average, and most for the least experienced, who learned faster with it. But that was an assistant alongside the work. Where the executor takes the routine cases entirely, the learning has to be designed. Common designs are rotating new staff through the exception queue, pairing them with junction holders, and having them review graded cases.

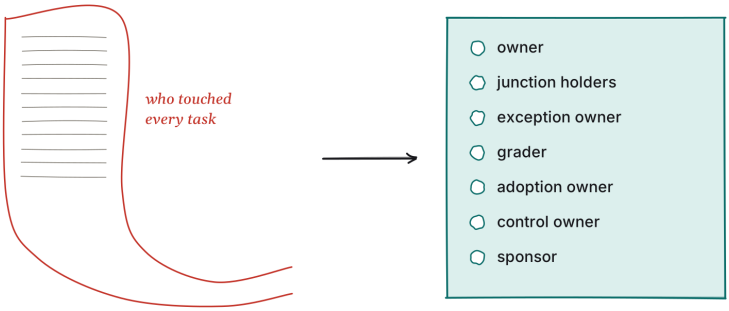


Like a ladder whose bottom rungs are taken away: *no one can climb to the hard top rungs unless a new way up is built.*

Loop roles and the RACI chart

Many companies assign responsibility per task with a RACI chart, listing who is responsible, accountable, consulted and informed. With an executor doing

many of the tasks, per-task charts become long and misleading. Embedded loops replace them with a short set of loop roles: owner, junction holders, exception owner, grader, adoption owner, control owner and sponsor. Each is a named person with a defined decision.



Like swapping a receipt as long as your arm for a team sheet: *seven named roles, each with its own decision.*

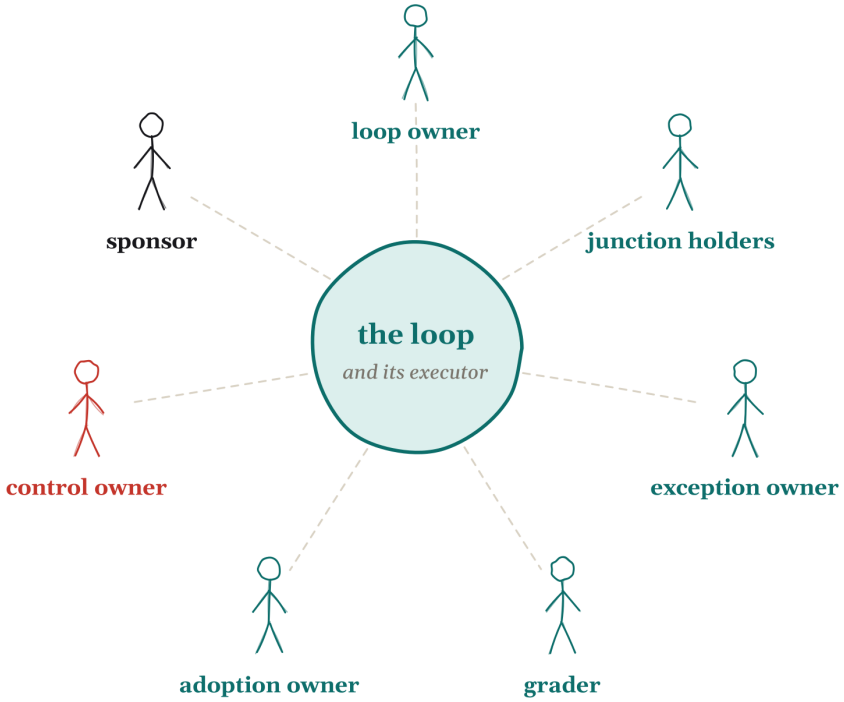
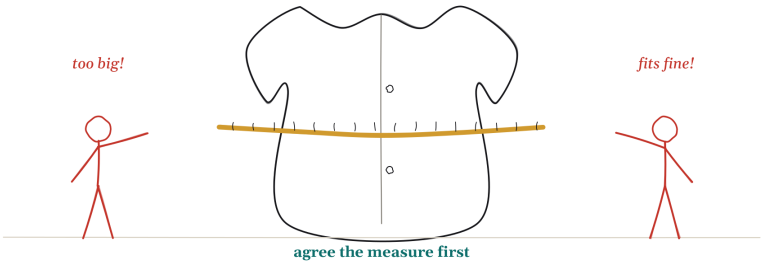


FIGURE 23 People and accountability. Every loop has named people around it: the owner of its outcome, the holders of its junctions, the owner of its exceptions, its grader, the owner of adoption, the control owner who signs its limits and the sponsor above them all.

XXIV

Economics

This area covers what an embedded loop costs, what it saves and when, and how those numbers are made credible. Most disputes about the value of AI are disputes about measurement. This area defines the measurements so the disputes can be settled.

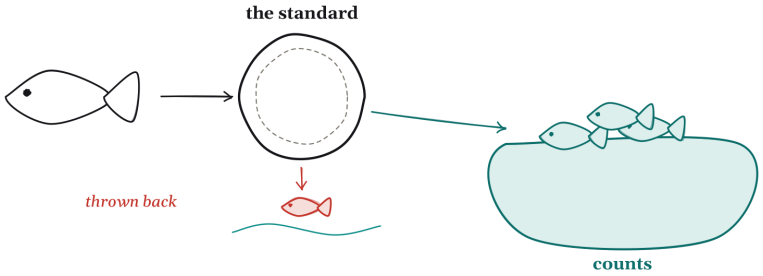


Like a tape measure in an argument about fit: *agree first how the jacket is measured, and the dispute about value settles itself.*

Where it comes from

This area measures whether the closed loop is worth running. Its foundations are accounting and measurement theory, including the warning that a measure turned into a target stops measuring. Cost per accepted outcome ties the economics to the standard, so that only results which passed the grader count.

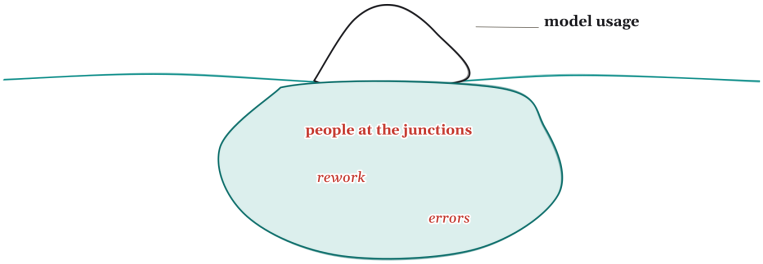
Concept	Kind	First used at
Fully loaded cost	Concept	Map
Baseline	Instrument	Map
Cost per accepted outcome	Measure	Own
Body and tail economics	Measure	Own
Economics sheet	Instrument	Own
The value curve	Concept	Read, Own
The capacity to embed	Concept	Scale
Measures that mislead	Concept	All steps
Pricing by outcome	Concept	Design



Like a fishing gauge: *only fish that pass the measuring ring count toward the catch, so the economics count only results that met the standard.*

Fully loaded cost

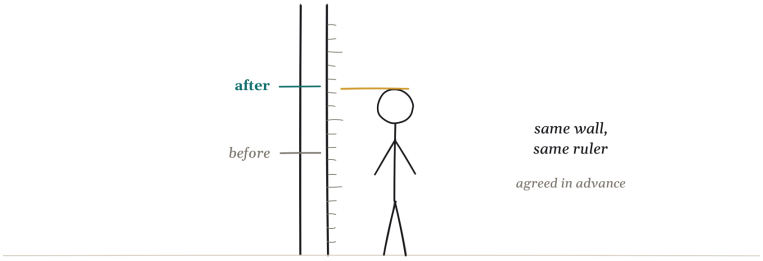
Fully loaded cost counts everything a loop consumes. That includes people's time at its real cost, systems and platforms, model usage, grading and audit, rework, and the cost of errors that reach customers or suppliers. Partial counts flatter whichever side leaves the most out. The commonest omission in embedded loops is the people still working inside them, at the junctions and in the exception queue.



Like an iceberg: *the model bill shows above the water, while people's time at the junctions, rework and errors sit below it.*

Baseline

The baseline is the loop's measured performance before it changes, defined in the Work and Process area. Its economic role is to be the reference against which every saving is claimed. It is measured the same way as the later economics, on a comparable period, and finance agrees the method in advance.

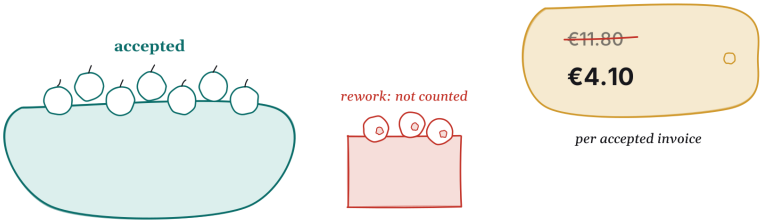


Like the pencil marks on a doorframe: *the child is measured the same way each time, against the same wall, so the growth claimed is real.*

Cost per accepted outcome

Cost per accepted outcome is the fully loaded cost of a loop over a period, divided by the outcomes in that period that met the standard. It is the central economic measure of an embedded loop. Dividing by accepted outcomes, not by cases processed, means that a loop which processes cases quickly but produces rework is not counted as cheap.

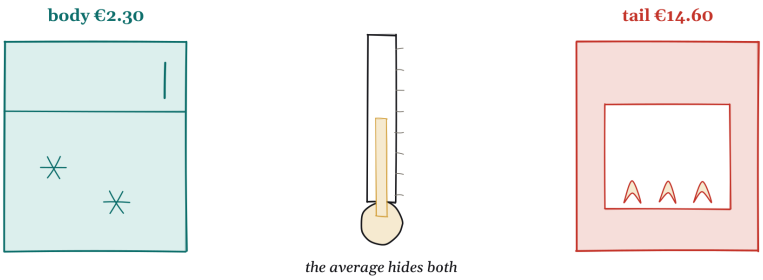
At the manufacturer, the baseline was €11.80 per invoice. After embedding, it was €4.10 per accepted invoice, measured over eight weeks across a quarter-end. At 7,400 invoices a month, the difference was worth about €680,000 a year.



Like paying a picker per good apple, not per apple picked: *divide the cost by the ones that passed, so fast but bruised picking is not cheap.*

Body and tail economics

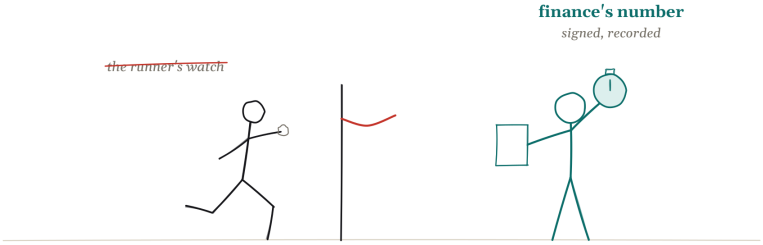
Cost per accepted outcome is reported separately for the body and the tail. A loop can cut the cost of routine cases sharply and leave the tail almost unchanged, because people still decide there. The average hides both facts. At the manufacturer, body invoices cost €2.30 each and tail invoices €14.60.



Like reading one thermometer between a freezer and an oven: *the average looks comfortable and describes neither, so body and tail are reported apart.*

Economics sheet

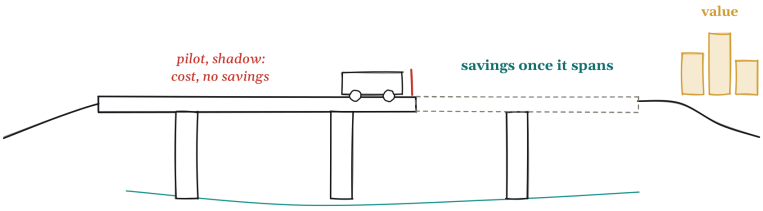
The economics sheet records the cost lines, the accepted outcomes, the cost per accepted outcome for the body and the tail, the comparison with the baseline, and the period. It is signed by finance, usually the planning and analysis lead, at Step 8. It is finance's number, not the programme's.



Like the official timekeeper at a race: *the runner's own watch may read faster, but the recorded time is the timekeeper's, as the economics sheet is finance's.*

The value curve

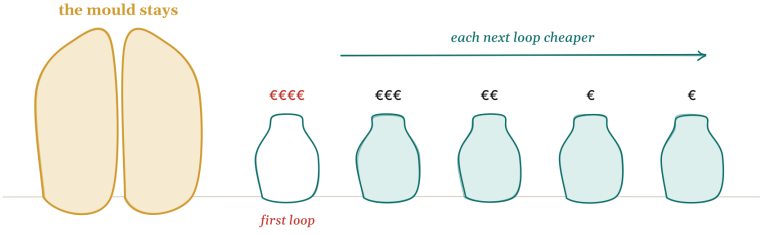
The value of an embedding programme arrives late. Pilots and shadow running produce evidence but no savings, because people still do all the work. Savings begin at Bound, on the body of the distribution, and grow at Core. A programme that stops before Bound has spent most of its cost and captured almost none of its value. This shape explains why so many surveys find little return from AI, and why finishing matters more than starting.



Like building a bridge: *the money goes into the piers first, and no one crosses until the span reaches the far bank, so stopping halfway earns nothing.*

The capacity to embed

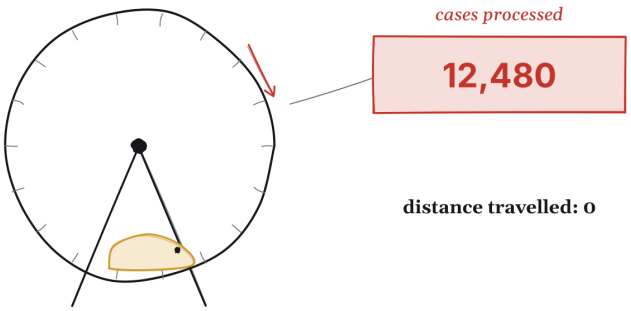
The first loop through the lifecycle is the most expensive, because the company is learning the steps. Later loops are cheaper and faster. The lasting asset is not any single loop's savings. It is the company's capacity to move loop after loop through the lifecycle, measured by the falling time and cost per loop.



Like a potter's mould: *the first pot pays for the mould, every later pot comes quicker and cheaper, and the mould is what you keep.*

Measures that mislead

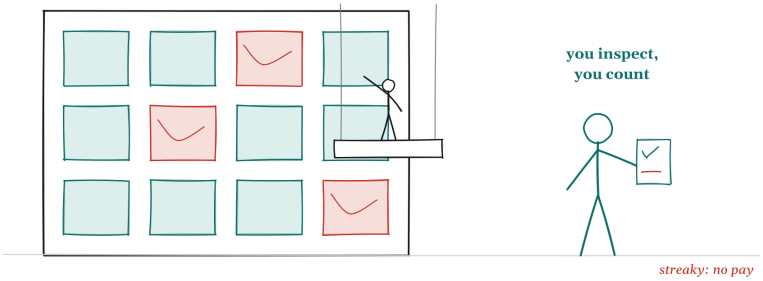
Several popular measures rise whether or not the business is better off: cases processed by AI, agents deployed, steps automated, hours saved estimated from task timings. They count activity, not accepted outcomes. The economist Charles Goodhart observed that a measure that becomes a target tends to stop measuring what it was meant to. Activity counts are especially prone to this, because they are easy to raise. They should be kept out of embedding reports.



Like a hamster wheel with a mileometer: *the count climbs fast while the hamster goes nowhere, so activity counts stay out of the report.*

Pricing by outcome

Some vendors price their executors by the outcome rather than by licence or usage: per resolved ticket, per processed invoice. Outcome pricing aligns the vendor with the loop, but only if "outcome" means an outcome that met the company's standard, graded by the company. A per-outcome price on outcomes the vendor counts itself is a usage price under another name.



Like paying a window cleaner per clean window: *fair only if you, not the cleaner, inspect which windows are clean.*

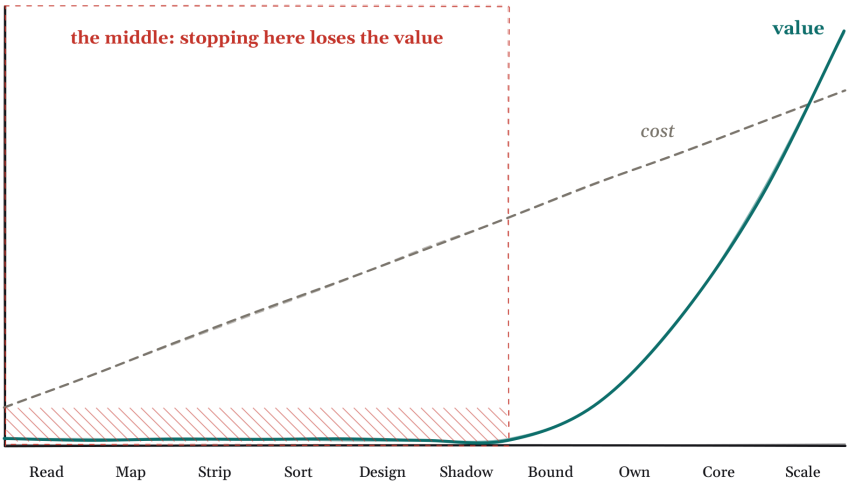
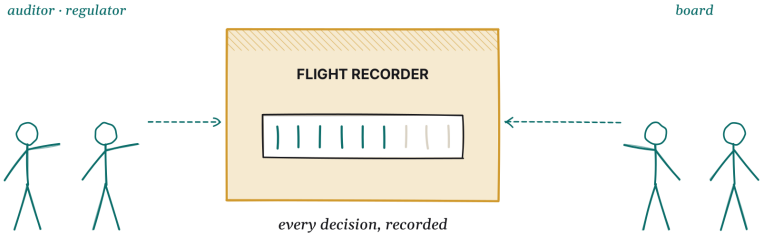


FIGURE 24 Economics. Cost accumulates from the first step; value arrives when the executor acts. Stopping before bounded running spends most of the cost for almost none of the value.

XXV

Governance

This area covers how the decisions made in embedding are recorded, reviewed and connected to the company's obligations. It is the area that turns good practice into evidence an auditor, a regulator or a board can rely on.

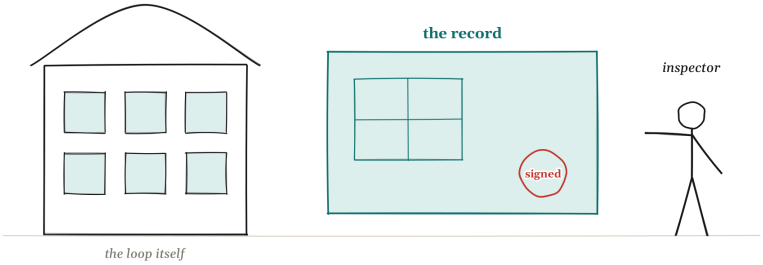


Like a flight recorder: *each decision is written down as it happens, so an auditor, a regulator or a board can rely on it later.*

Where it comes from

This area records the symbolic layer so that people outside the loop can rely on it. Signatures turn agreement into accountability. The record set is the model of the loop that auditors, regulators and boards read.

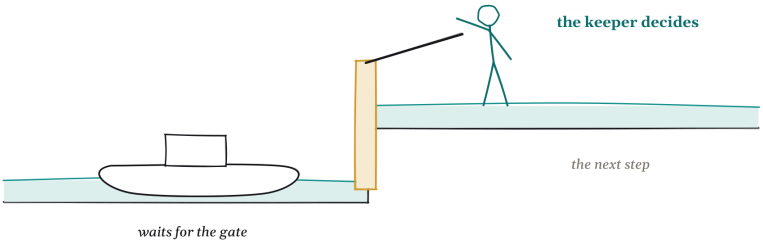
Concept	Kind	First used at
Gate	Concept	All steps
Gate review record	Instrument	Bound
Operating review	Instrument	Core
The record set	Concept	All steps
Scope change	Concept	Core
Governance frameworks	Concept	Bound, Core
High-risk loops	Concept	Read, Sort, Bound
Liability for what the executor says	Concept	Design



Like a building's stamped drawings: *inspectors read the signed plans to learn how the building works, without taking down a wall.*

Gate

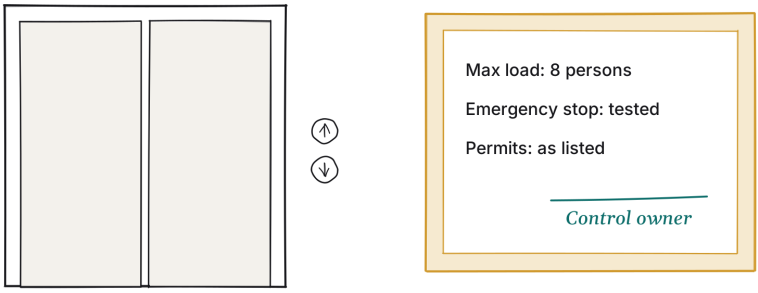
A gate is the point at the end of a lifecycle step where a named person decides whether the work may move on. It has written criteria and a signer. The idea comes from stage-gate reviews in product development, described by Robert Cooper in 1990, where gates protected companies from spending heavily on projects that should have been stopped. Gates in embedding protect against something different: an executor acting before the business has decided what it may do.



Like a canal lock: *the boat rises to the next level only when the keeper, having checked the water, opens the gate.*

Gate review record

The gate review record documents the decision at Bound. It records the limits the executor may act within, the evidence that supported them, the permissions granted, the switch-off test and any audit notes. It is signed by the control owner. It is the document an auditor asks for first.

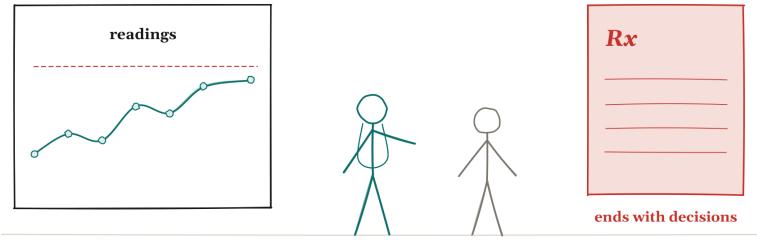


the first page an auditor asks for

Like the inspection certificate inside a lift: *it states the load limit and the last emergency-stop test, signed by the inspector, and it is checked first.*

Operating review

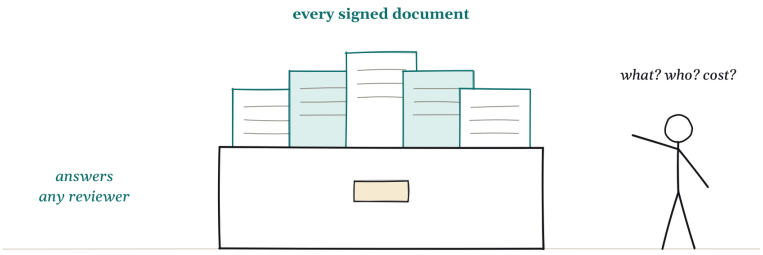
The operating review is the monthly review of a loop in the core, held by the loop owner. It reads the share of outcomes meeting the standard for the body and the tail, the cost per accepted outcome, the exception queue, the junction override rate and any drift. It ends with recorded decisions. A review that ends without decisions is a status meeting.



Like a doctor's check-up: *readings are taken every month, and the visit only counts if it ends with something prescribed.*

The record set

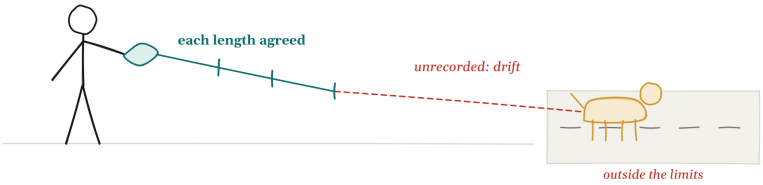
The record set is the full collection of signed documents for a loop. It includes the map, baseline, strip register, rule and decision registers, contested-decision log, canvas, junction register and standard. It also includes the shadow report, gate review record, loop charter, economics sheet, operating reviews and switch-off plan. Together they answer the questions any reviewer will ask: what the loop does, under which rules, who decides what, how it is graded, what it costs, and who answers for it.



Like a house's deeds box: *the survey, plans, permits and receipts kept together answer any question a buyer could ask about the house.*

Scope change

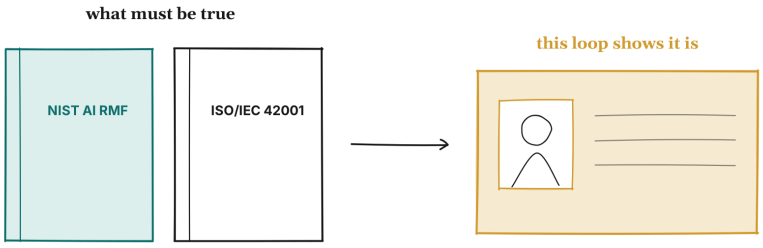
A scope change is any widening of what the executor may do alone: a new value band, a new kind of case, a new counterparty. Each is a small gate, recorded and agreed by the control owner. Unrecorded scope changes are the commonest way a loop drifts outside the limits that were signed.



Like an extending dog lead: *each extra length is clicked out on purpose, and a lead left unlocked lets the dog drift into the road.*

Governance frameworks

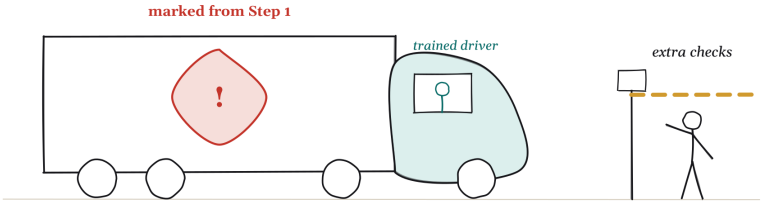
Several frameworks set out what good AI governance requires. The NIST AI Risk Management Framework, published in January 2023, organises the work into four functions: govern, map, measure and manage. ISO/IEC 42001, published in 2023, is a certifiable management-system standard for AI. Both describe what must be true. The record set is how a particular loop shows that it is.



Like the rules of the road and a driving licence: *the frameworks say what a safe driver must do, and the record set shows this one passed.*

High-risk loops

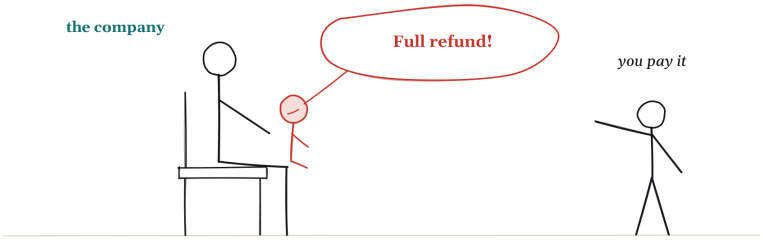
Some loops fall within laws on automated decisions. The European Union's AI Act, adopted in 2024, treats certain uses as high risk, such as decisions on employment, creditworthiness and access to essential services. It requires risk management, logging, human oversight by competent people and monitoring by deployers. Loops in these areas need legal or compliance input at Sort and at the Bound review. Their junction registers, switch-off plans and charters serve as legal evidence. A loop's risk category should be noted on the map at Step 1.



Like a lorry carrying hazardous goods: *it wears its warning placard from the start, and gets extra checks and a trained driver on the way.*

Liability for what the executor says

A company is generally held to what its executor tells customers and suppliers. In 2024, a Canadian tribunal held Air Canada responsible for incorrect refund information given by the chatbot on its website. Loops in which the executor makes promises, such as refunds, delivery dates and exceptions to policy, need those promises inside signed limits, with junctions where they exceed them.



Like a ventriloquist and his dummy: *whatever the dummy promises, the audience holds the ventriloquist to it, so promises stay inside signed limits.*



FIGURE 25 Governance. The record set of one loop: every signed document from the map to the operating reviews, each with its signer, shelved in lifecycle order.

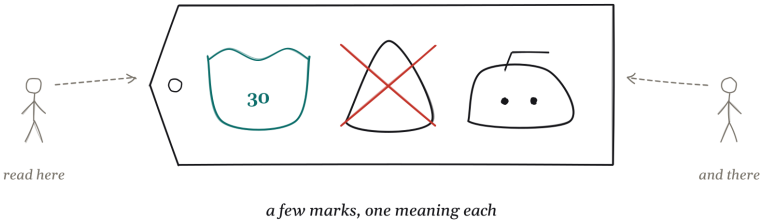
PART SIX

Across the Areas

XXVI

Notation

A discipline needs a shared way of drawing its instruments, so that a loop drawn in one company can be read in another. This chapter defines the notation used for the Work Loop Canvas, the Work Loop Map and the process map. It is deliberately small. Every symbol stands for a concept defined in the areas, and nothing is drawn that does not carry meaning.



Like the care symbols on a clothing label: *a handful of small marks, each with one fixed meaning, read the same way in any country.*

States

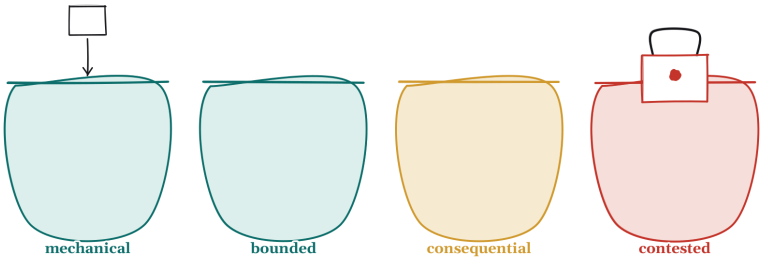
Every node of a canvas, and every document in the record set, is in one of three states. Blank means nothing has been written. Drafted means a proposal exists but nobody has signed it. Signed means a named person has accepted it, with a date. On drawings, blank is an outline in ink, drafted is filled in amber, and signed is filled in teal. Lines between signed nodes are drawn solid in teal. Lines where either end is only drafted are dashed in amber. The rule is simple: colour is earned by signature, not by effort.



Like a cheque: *blank, then filled in, then signed, and only the signed one can be cashed. Colour on a drawing is earned the same way.*

Decision classes

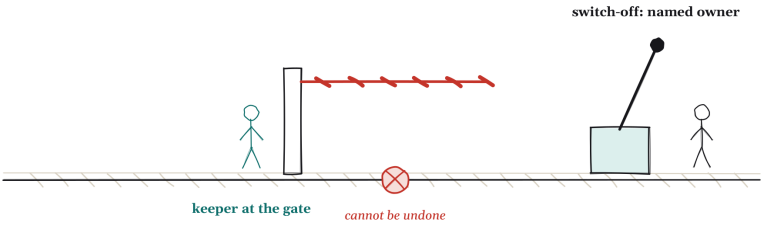
The four decision classes are drawn as bins. Mechanical and bounded decisions are teal, consequential decisions amber, and contested decisions red, with a lock on the bin while any contested decision is open.



Like a row of sorting bins: *routine items drop straight in, some wait for a check, and one bin stays padlocked while a dispute about it is open.*

Junctions and switches

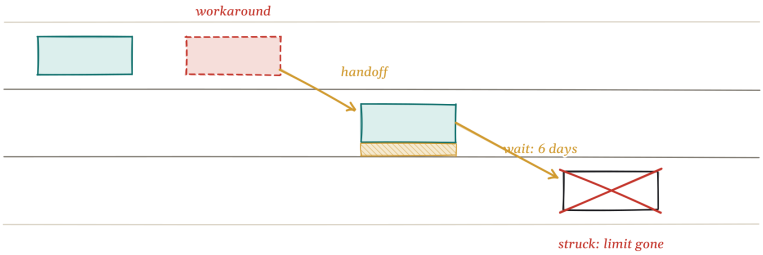
A junction is drawn as a gate with a person beside it. A red valve on a line means an action that cannot be undone and therefore requires a junction. The switch-off is drawn as a lever, and the person who holds it is named beside it.



Like a level crossing: *a keeper stands at the gate where a mistake cannot be undone, and one named person holds the lever that stops everything.*

Process maps

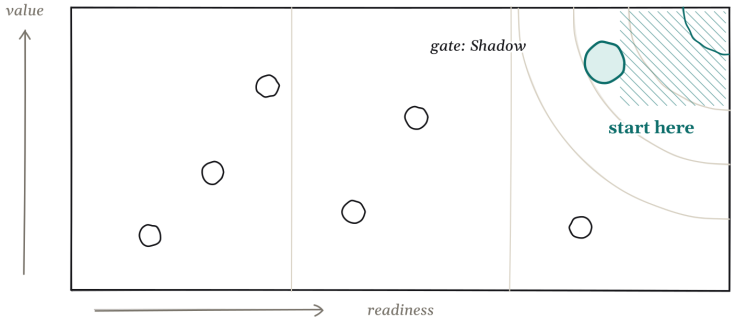
On process maps, steps are rectangles in the lane of whoever does them. A workaround, which is a step that compensates for a constraint, is drawn with a red dashed outline. Rework has an amber dashed outline. Waits are drawn as hatched amber blocks beneath the step they follow, with their duration. Handoffs between lanes are drawn as solid amber arrows. Struck steps are crossed through in red, with the constraint written beside them.



Like a running track seen from above: *each runner keeps a lane, the baton passes between lanes, and anyone stopped or detouring shows up at once.*

The map

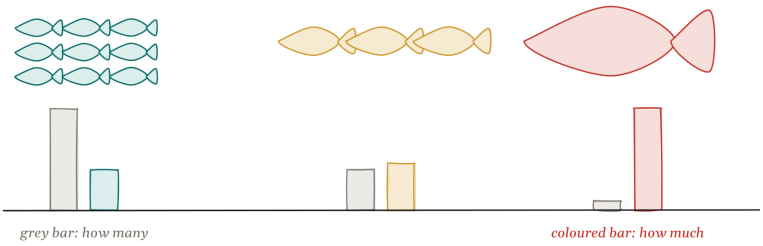
On the Work Loop Map, each loop is a circle placed by value to the core (vertically) and readiness (horizontally). The upper right region, where to start, is tinted teal. Handoffs between loops are lines. The loop currently in a lifecycle is drawn larger, with its gate written beside it.



Like a hiking map with pins: *each loop sits where its value and readiness put it, the near summit is shaded as the place to start.*

Distributions

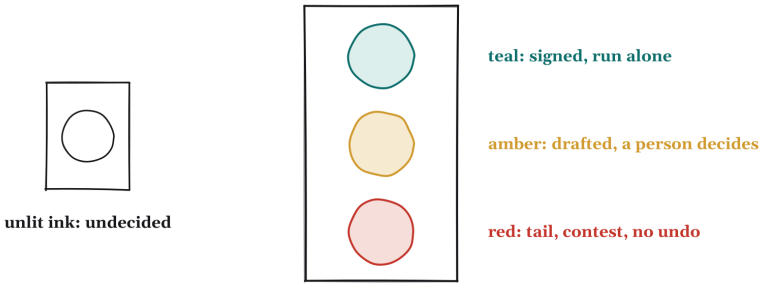
Distributions are drawn as paired bars for each band, grey for the share of cases and coloured for the share of value: teal for the body, amber for the middle and red for the tail.



Like a day's catch on the quay: *count the fish and weigh the fish, because a few big ones in the tail can outweigh the whole pile of small ones.*

Colour

The notation uses four colours, each with one meaning. Teal means signed, safe to run alone, or the body. Amber means drafted, waiting, or needing a person's decision. Red means a workaround, a contest, an irreversible action or the tail. Ink means everything not yet decided. Drawings should never use colour for decoration.



Like a traffic light: *green means go alone, amber means a person must decide, red means stop, and an unlit lamp means nothing is decided yet.*

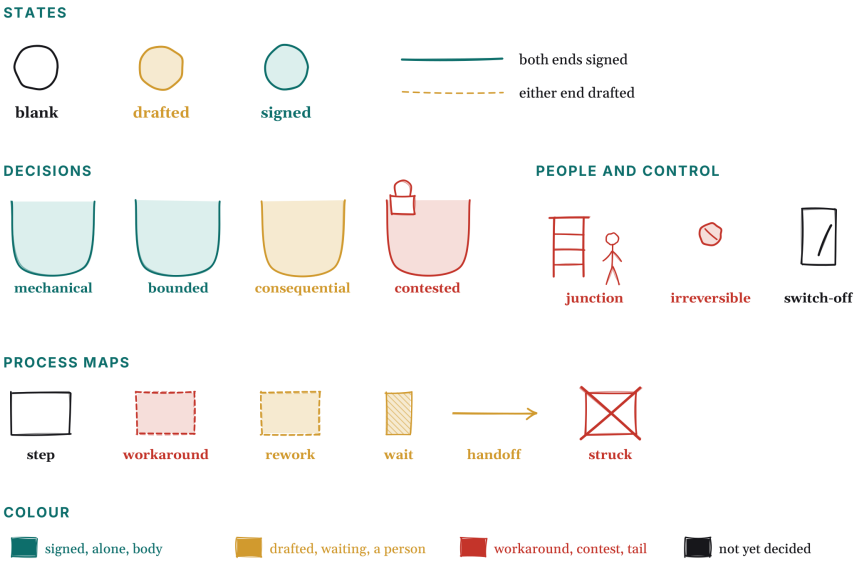


FIGURE 26 The notation. Three states earned by signature, four decision classes, the junction and the switch, the marks used on process maps, and the four colours with one meaning each.

XXVII

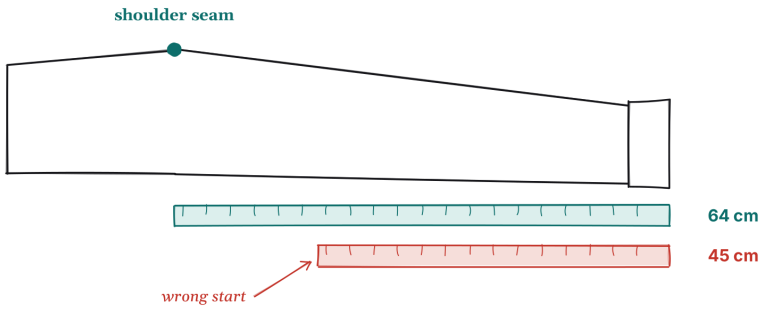
The Measures

This chapter gives the exact definition of every measure used in the discipline. Measures that sound alike are often calculated differently from one company to the next. Comparing them is then meaningless, and arguments about results become arguments about arithmetic. Each definition below states what is counted, how, over what period, and the most common way it is miscalculated.

Measure	Definition	Common miscalculation
Share of cases, by band	Cases in the band divided by all cases in the period	Bands chosen after looking at the data to flatter a story
Share of value, by band	Value of cases in the band divided by the value of all cases	Using list values rather than actual amounts
Concentration	Share of value carried by the top 5% of cases by value	Computed on a period too short to contain the tail
Value to the core	Score from 1 to 5, agreed in session, recorded with reasons	Scored by how impressive the AI would look
Readiness	Score from 1 to 5 on known rules, measurable outcome, available context, willing owner, settled decisions	Scored once and never updated
Volume	Cases arriving per period	Counting batches or documents instead of cases
Fully loaded cost per case	All costs of the loop in the period, including people at real cost, systems, rework and errors, divided by cases	Leaving out people in junctions and queues, or rework
Elapsed time	Time from a case's arrival to its accepted outcome, as a median and a 90th percentile	Starting the clock at first touch rather than arrival
Hands-on time	Minutes of work done on a case, summed across steps	Estimated from interviews rather than sampled
Flow efficiency	Hands-on time divided by elapsed time	Mixing averages of different case samples
Exception rate	Cases that left the normal path, divided by all cases	Excluding cases resolved informally
First-time-right	Cases accepted without rework, divided by all cases	Counting a case as right if the error was caught internally
Strip ratio		

Measure	Definition	Common miscalculation
	Steps struck divided by steps on the signed map	Counting struck steps without named constraints
Contradictions	Pairs of rules that cannot both be followed, found and decided	Recording them as noise rather than findings
Contested decisions open	Contested decisions logged and not yet settled	Reported as zero because nobody asked
Agreement, body and tail	Cases where the executor's result matched the accepted result, divided by cases, per segment	Reporting one overall figure
Bounded share	Cases, and separately value, inside the executor's signed limits, divided by all	Reporting cases only, never value
Standard met	Outcomes meeting the standard, divided by all outcomes, per segment	Grading a sample rather than every outcome
Cost per accepted outcome	Fully loaded cost in the period divided by outcomes meeting the standard, per segment	Dividing by cases processed
Override rate	Junction decisions where the holder overruled the executor's evidence, divided by junction decisions	Treating zero as success
Exception age	Age of the oldest case in the exception queue	Reporting the average instead of the oldest
Time per step	Elapsed weeks in each lifecycle step, per loop	Measuring only the build, not the decisions

Two rules apply to every measure. Report the body and the tail separately wherever a measure can differ between them. And never compare a measured figure with an estimated one.



Like a tailor's tape: *measure a sleeve from a different starting point and the number changes, so every measure states exactly where counting starts.*

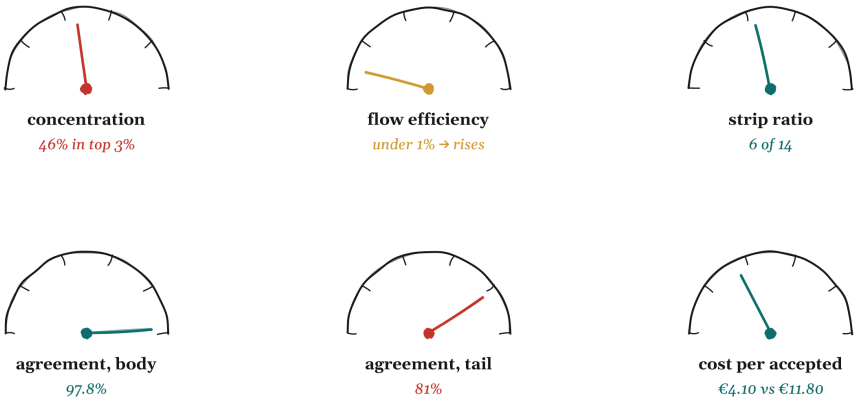
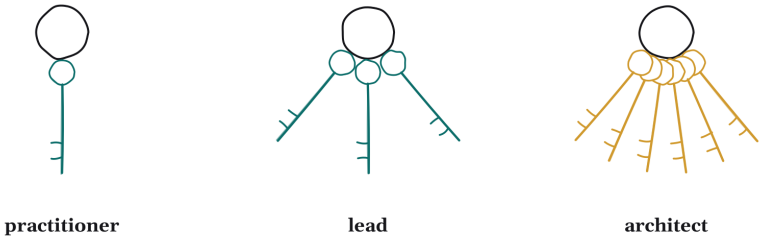


FIGURE 27 The measures that matter, read on one panel: concentration, flow efficiency, strip ratio, agreement for body and tail, cost per accepted outcome and override rate. Activity counts are left off on purpose.

XXVIII

The Competencies

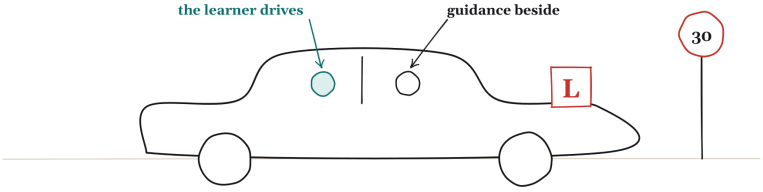
A discipline is held by people. This chapter describes what a business engineer needs to know and be able to do, in three levels. The levels describe what a practitioner can be trusted with, not years of service.



Like key rings of different sizes: *each level is trusted with more doors, and the ring grows with what someone can be trusted to open, not their years.*

The practitioner

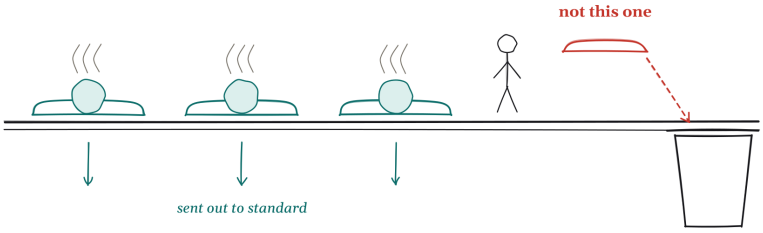
A practitioner can run the steps of a lifecycle for a loop under guidance. They can draw a process map with operators, walk real cases through it, and measure a baseline. They can gather rules from documents, systems and interviews into a register, and find contradictions. They can sort decisions into classes and recognise a contested one. They can prepare every document in the record set to the standard required for signature. They know what each gate requires and who signs it.



Like a learner driver with an instructor beside them: *they drive every part of the route themselves, under guidance, and know every rule of the road.*

The lead

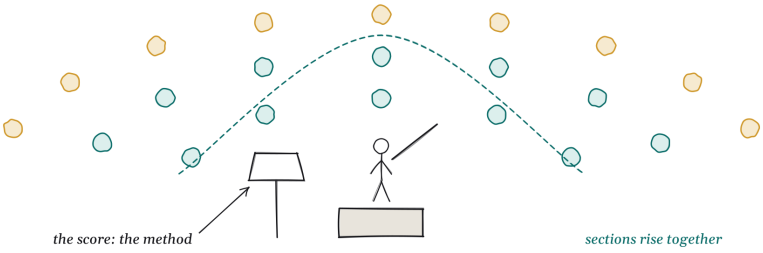
A lead can run a lifecycle end to end, and run several at once. They can read a company's loops, facilitate the scoring of the map, and recommend a first loop the sponsor will accept. They can design junctions with real evidence and reversal, write a standard that grades outcomes, and set bounded limits from shadow evidence. They can build an economics sheet that finance will sign. They can lead the contested-decision conversation with senior people. And they can tell when a loop should not proceed.



Like a head chef at the pass: *several dishes going at once, each sent out to standard, and the judgement to bin one that should not go.*

The architect

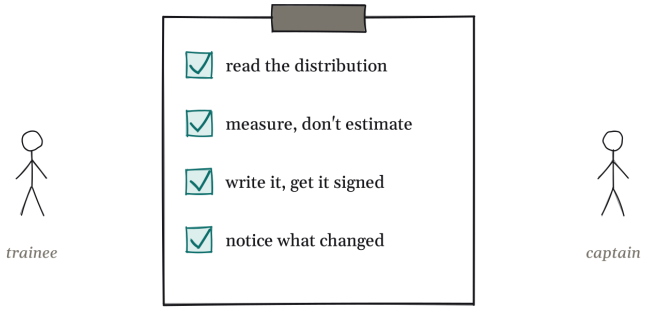
An architect holds the method for a company. They maintain the instruments and the notation, train practitioners and loop owners, and run the business engineering function. They read the map as a whole, see which chains must scale together, and shape the role change plan with the sponsor. They connect the inside view of loops with the outside view of the business model. They also connect the record set with the company's governance obligations.



Like a conductor holding the score: *they keep the method, train the players, hear the whole orchestra and know which sections must rise together.*

What all three share

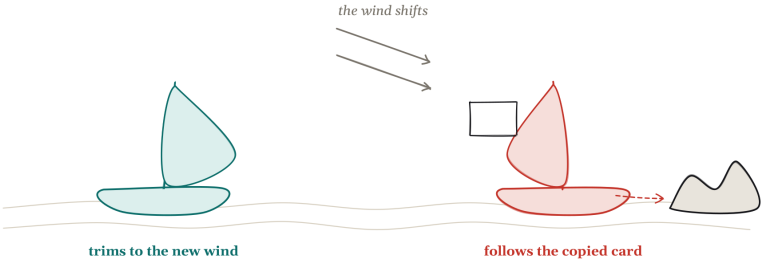
Every level shares four habits. Read the distribution before designing anything. Measure rather than estimate. Write down decisions and get them signed. And notice when the world has changed. These are the habits of the method loop, practised on the work of embedding.



Like a pilot's pre-flight checklist: *the trainee and the captain run the same four checks every time, however senior they are.*

What they need to know about the foundations

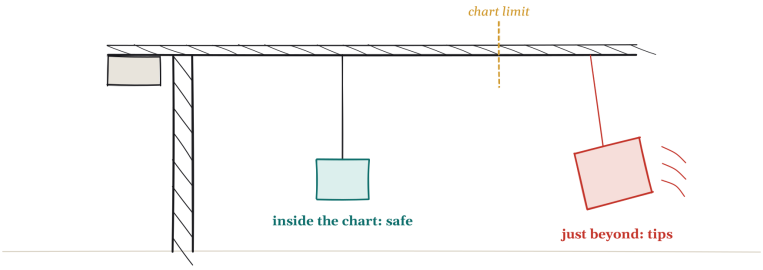
Every level also needs a working grasp of the foundations in Part One. That does not mean mathematics. It means understanding why rules are precise but incomplete, and why learned models are flexible but come without guarantees. It means seeing how the two are combined, and why control requires a model and a closed loop. A practitioner who understands those four ideas will make sound design choices in situations no book anticipated. One who does not will copy templates.



Like a sailor who understands the wind: *when it shifts, they trim for the new wind, while one following a copied route card drifts onto the rocks.*

What they need to know about the technology

A business engineer does not need to build an executor. They do need to know its parts well enough to make the decisions those parts require: the model, instructions, context, actions, platform and grader. They should also know the uneven frontier of what current executors do well and badly, and how that frontier is tested. A business engineer who cannot tell what an executor can and cannot do will either over-trust it or design it out of the work.



Like a site manager reading a crane's load chart: *no need to build the crane, but lifts inside the chart are safe and a load just beyond it tips.*

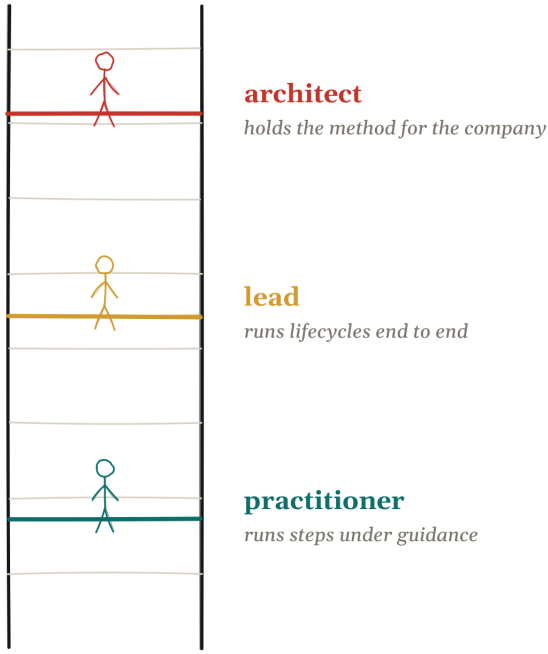
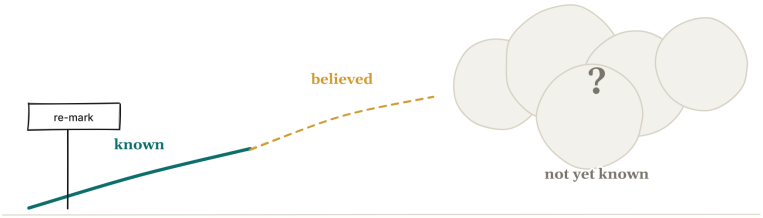


FIGURE 28 The competencies. Three rungs of practice: the practitioner runs steps under guidance, the lead runs whole lifecycles and several at once, the architect holds the method for the company. All three share the habits of the method loop.

XXIX

The Evidence Base, as Read in 2026

A body of knowledge should say what is known, what is believed, and what is not yet known. This chapter is dated, and it should be reread as the evidence changes. The concepts in the areas are designed to outlast the specific findings below.



Like a hill trail: *a marked path where the ground is known, a faint track where it is believed, then fog, and the markings are redone as the ground shifts.*

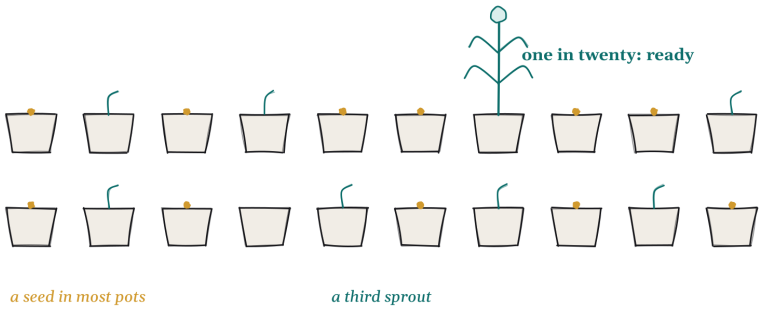
What is reasonably established

Most companies use AI somewhere, and few have embedded it into core processes at scale. McKinsey's 2025 survey found 88% of organisations using 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. MIT's 2025 study found that only about 5% of custom enterprise AI tools reached production.

AI assistance raises productivity on many tasks, unevenly. Studies of customer support and of consultants found substantial gains on tasks inside the frontier of what models do well, and losses on tasks just outside it, where people trusted answers they should have checked.

People supervising automated systems tend to over-trust them and lose skills over time. Research on automation stretching back to the 1980s documents this consistently.

Process documents diverge from how work actually runs. Process mining of event logs has shown this repeatedly across industries.



Like a seed tray: *almost every pot has a seed in it, about a third have sprouted, and only one in twenty is a plant ready for the ground.*

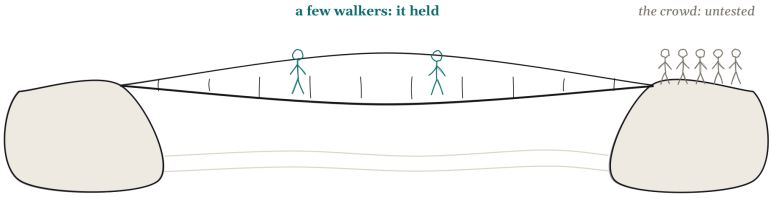
What is believed but less tested

That separating decisions from process, and sorting them into classes, is the main lever for embedding executors safely. Practitioners report it, and early research on combining agents with explicit process rules supports it. Large controlled comparisons are still rare.

That combining learned models with explicit rules and verification makes systems more reliable than either alone. Demonstrations in mathematics and early research on agents that follow explicit process rules support it. Systematic evidence from enterprise deployments is still thin.

That value arrives late, at bounded running and after, rather than at the pilot. The shape is consistent with the survey evidence on stalled programmes. It has not been measured directly across many companies.

That the second and later loops take much less time than the first. Practitioners report it, and published longitudinal data is thin.

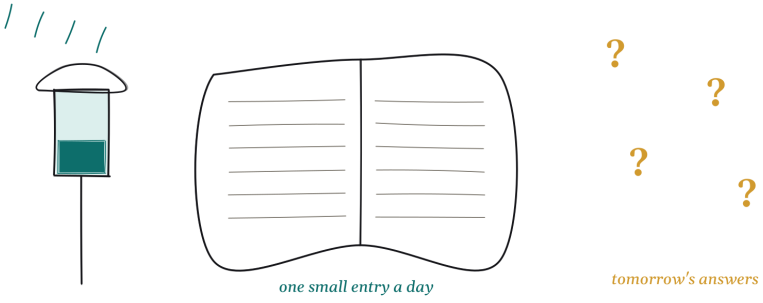


Like a new footbridge crossed by a few walkers: *it held every time so far, but nobody has yet run the full load test with a crowd.*

What is not yet known

How quality changes over years in loops where the model is replaced several times. How the skills needed at junctions can best be rebuilt when routine cases no longer teach them. And how regulators will interpret human oversight requirements for executors that decide in practice, as enforcement begins.

These open questions are the research agenda of the discipline. Practitioners who record their lifecycles, with their measures, are generating the evidence that will answer them.

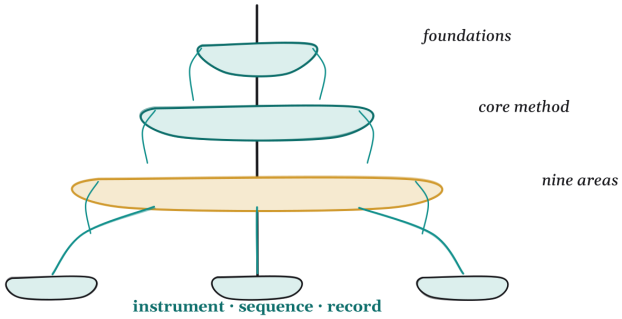


Like farmers keeping weather diaries: *each day's entry is small, but together the records will answer questions nobody can answer yet.*

XXX

How the Discipline Cascades

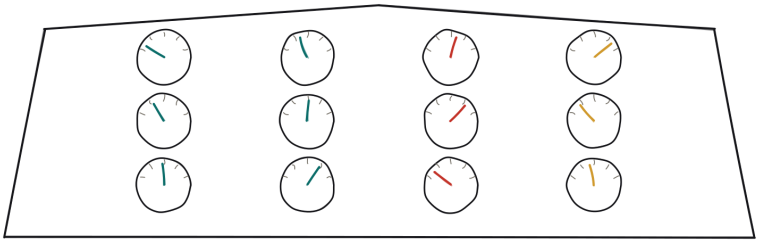
Everything else in the discipline follows from this book. The foundations explain why an embedded loop must be a neurosymbolic, closed-loop system. The core method explains how a practitioner builds the model that regulates it. The nine areas define the knowledge that model is made of. From there the discipline cascades into three working forms.



Like a tiered fountain: *water poured in at the top fills each basin in turn, and the bottom basin spills into three working spouts.*

The instrument

The Work Loop Canvas is the nine areas compressed onto one page for one loop. Its twelve nodes are the areas made concrete. The outcome and economics come from value and economics, and the compensations from work and process. The rules and decisions come from the symbolic layer. Context and actions come from the executor, and junctions, exceptions, the standard and handoffs from control. The people come from accountability. The Work Loop Map is the same knowledge at the level of the company.

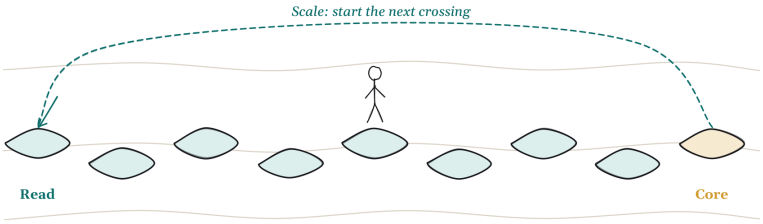


twelve dials, one loop, one panel

Like a car's dashboard: *everything under the bonnet shows up as one dial each, so the whole car is read from one panel.*

The sequence

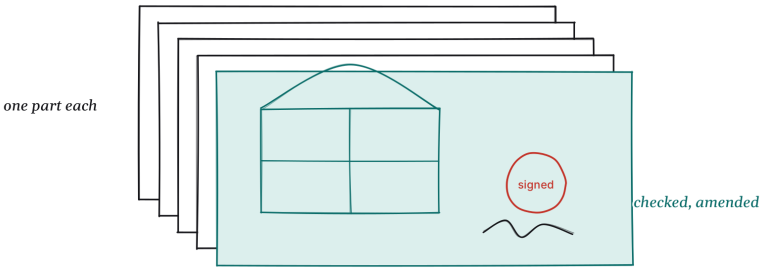
The embedding lifecycle is the core method run in time, one loop at a time. Read is the distribution and the map. Map, Strip and Sort are abstraction and compression applied to real work, producing the symbolic layer. Design closes the loop on paper. Shadow and Bound are testing: falsifiable claims about agreement and limits, put to real cases. Own measures the economics and names the people. Core runs the closed loop. Scale re-reads the distribution and starts again.



Like stepping stones across a river: *one loop crosses stone by stone from Read to Core, and at Scale you turn back to start the next crossing.*

The record

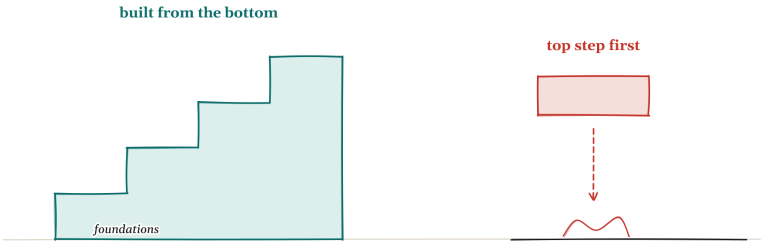
The template kit is the symbolic layer written down in a form people sign. Each template records one part of the model that regulates a loop. These include the map, baseline, registers, contested-decision log, junction register and standard. They also include the shadow report, gate review record, charter, economics sheet, operating review and switch-off plan. Together they are the record set: the company's model of its own work, auditable and revisable.



Like a building's set of signed drawings: *each sheet records one part of the house, and together they can be checked and amended.*

Why the order matters

The cascade runs one way. A company that adopts the templates without the areas will fill in forms it does not understand. A company that adopts the areas without the core method will define terms without reading its own distribution. And a company that adopts the method without the foundations will not know why the symbolic layer, the human junctions and the closed loop are not optional. Starting from the foundations is slower for a week and faster for every loop after.



Like a concrete staircase: *each step rests on the one below, so laying the top step first leaves it with nothing to stand on.*

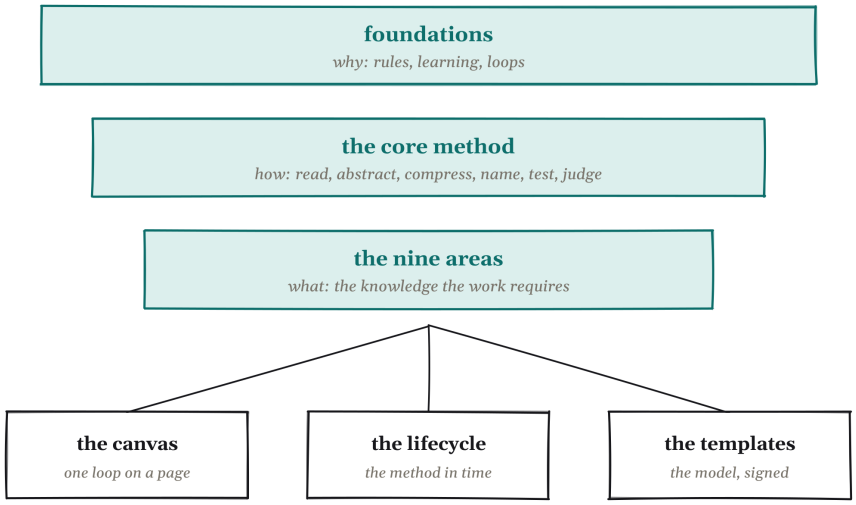


FIGURE 29 How the discipline cascades. The foundations explain why; the core method explains how; the nine areas define the knowledge; the canvas, the lifecycle and the templates follow.

PART SEVEN

The Mental Models Library

XXXI

How the Models Library Works

A mental model is a compression with a name. It takes a pattern that recurs, states it in a sentence, and lets a practitioner recognise it the next time it appears. Earlier parts of this book introduced many such models where they first mattered. This part gathers them into a library a practitioner can consult.

The library is organised by the type of loop in which each model matters most, not by discipline of origin. A practitioner rarely asks which models come from statistics. They ask what to watch for in the loop in front of them. A loop where a few customers carry most of the money needs different models from a loop that screens job candidates, and both need different models from a loop that runs through outside carriers.



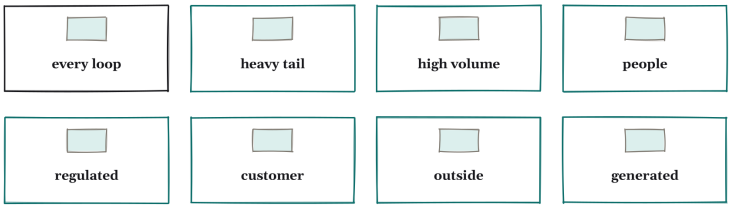
Like a field guide arranged by habitat rather than by family: *you look up what lives where you are standing, not where the species came from.*

Eight types of loop

The first chapter holds the models that apply to every loop, whatever its shape: the body and the tail, the contested decision, the junction, requisite variety, the good regulator and the others that make up the core of the discipline.

Seven chapters follow, one for each type of loop. Heavy-tailed loops are those where a few cases carry most of the value or risk. High-volume routine loops are dominated by a large body of similar cases. Loops that decide about people affect individuals′ work, pay or credit. Regulated and audited loops are read line by line by auditors or regulators. Customer-facing loops are judged by the customer. Loops that depend on outside parties run through suppliers, carriers or banks. Loops whose steps are generated let an executor find its own path to an outcome.

The last chapter covers loops in a chain, because every loop hands work to another, and a loop's tail is often the next loop's exception.



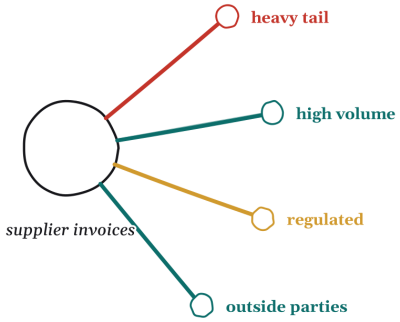
Like a hospital's wards: *the same body of medicine, but each ward keeps the instruments its patients need most close at hand.*

How to use the library

Most loops belong to more than one type. A supplier invoice loop is heavy-tailed, high in volume, audited and dependent on suppliers at once. Appendix F places each of the thirty-nine reference loops of *The Work Loop Library* by type, so a reader can find the nearest reference loop and read the chapters its marks point to.

Each entry follows the same order. It states the principle in plain words, says where the model comes from, shows how it appears in that type of loop, and says how to use it at a step of the lifecycle or in an instrument. It ends with the common misreading, because a model misapplied does more harm than a model never learned.

No single model is enough. As Charlie Munger argued, judgement comes from a latticework of models from many disciplines, each covering the others' blind spots. A practitioner who reads a loop through one model sees only what that model shows.



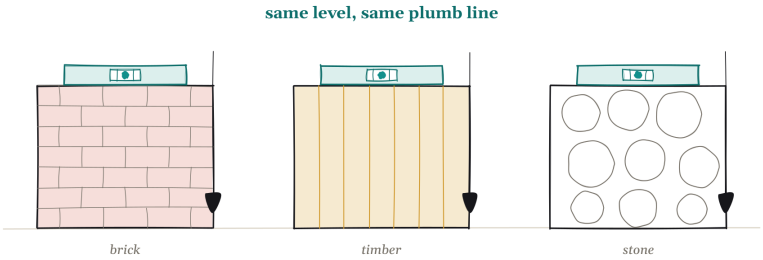
Like a key ring for a building: *one loop opens with several keys, because it belongs to several types at once.*

XXXII

Models for Every Loop

Some models belong to no single kind of work. They hold for a supplier invoice and a support ticket, for a candidate and a cash receipt. They describe what every loop is made of rather than what it is about. Every loop has an outcome, a spread of cases, rules and decisions, people placed at its edges, a cost, and some way of knowing whether it worked.

That is how to recognise this territory in a company: wherever those parts exist, these models apply. All 39 reference loops qualify, from Lead qualification and Supplier invoice approval to Candidate screening, Period close, Warranty claims and Shipment exception handling. These models matter most because the specialist models in later chapters rest on them. A team that reads its distribution, sorts its decisions and closes its loop has firm ground. A team that skips them will misapply everything built on top.



Like a builder's spirit level and plumb line: *the same two tools check every wall, whatever the house is made of, as these models check every loop.*

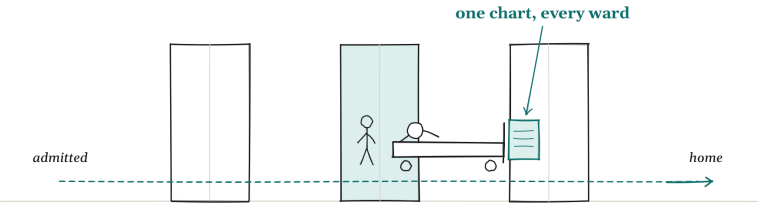
The Work Loop

Work is easiest to change when it is cut at the right size. Too small, and you automate a task while the waits around it stay. Too large, and nobody can own it. The right unit is one outcome, end to end, with the rules, decisions, people and costs that produce it.

Its ancestor is process reengineering. Michael Hammer and James Champy, in Reengineering the Corporation (1993), defined a business process as a set of activities that together create an output of value to a customer. The work loop itself is native to embedding practice. It lives in the Work Loop Canvas and on the Work Loop Map.

In Supplier invoice approval, the outcome is an invoice matched, coded, approved and posted as a liability due for payment. Work arrives from Purchase order issue and Goods and services receipt and leaves for the Payment run. No single team owns it all.

Use it at Read and Map. State the outcome so an outsider could accept or reject a result, then draw the loop's edges where work arrives and leaves. The common misreading is that a loop is one team's process, when that process is usually only a fragment of one.



Like a patient's chart clipped to the bed: *it travels from admission to discharge through every ward, so one outcome is followed end to end.*

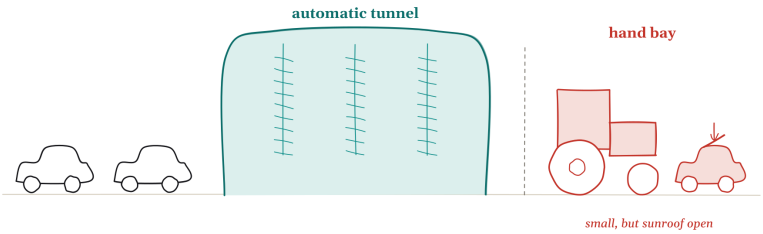
Body and Tail

In most loops, a large share of cases look alike, while a small share carry most of the value, risk or difficulty. The first group is the body and the second is the tail. They need different treatment. An average taken across both hides the difference.

The pattern has a long history. Vilfredo Pareto documented skewed distributions of income in his Cours d'économie politique (1896). Nassim Nicholas Taleb, in The Black Swan (2007), showed how rare, large cases dominate the results in heavy-tailed domains.

The manufacturer's invoices split cleanly. Invoices under €5,000 were 81% of the count and 14% of the spend. The 3% over €50,000 carried 46% of the spend. Shadow running told the same story from the other side. The executor agreed with the accepted result on 97.8% of body invoices and on 81% of tail invoices.

Use it at Read, then carry it through every later step. Report agreement, economics and drift separately for each part, and place junctions mostly in the tail. The common misreading is to draw the line by value alone, when a small first invoice from a new supplier with changed bank details belongs in the tail.



Like a car wash: *most cars roll through the automatic tunnel, while a tractor goes to the hand bay, and so does a small car with its sunroof open.*

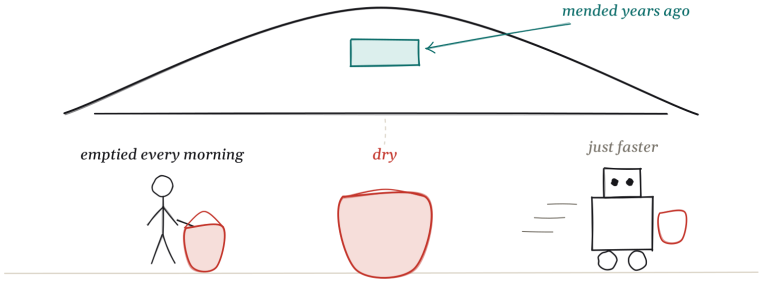
Compensations

Many steps in a process exist only because of a constraint that may have gone. A batch step waits for a system that once ran overnight. A second review survives from a year when errors were frequent. Over time, nobody can tell these workarounds from the real work.

Michael Hammer named the trap in "Reengineering Work: Don't Automate, Obliterate", published in the Harvard Business Review (1990). Automating old processes, he argued, mostly paves the cow paths. In embedding practice, the idea lives in the strip register and its summary measure, the strip ratio.

The manufacturer's invoice loop had a strip ratio of 43%. Almost half its mapped steps were workarounds for constraints that no longer applied. Attaching an executor to the loop as drawn would have run those steps faster. It would have left every wait between them in place.

Use it at Strip. For each step, ask which constraint it serves and whether that constraint still holds. Strike it if the constraint has gone, and record the constraint if not. The common misreading is that a capable executor will route around the workarounds by itself, when it simply runs them faster.



Like a pail still set under a roof leak mended years ago: *someone empties it every morning, and a machine would only empty it faster.*

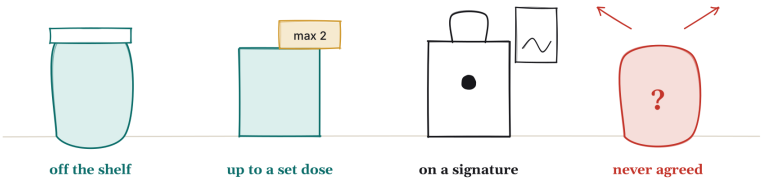
The Four Classes of Decision

Treating every decision alike is the fastest way to get embedding wrong. Some follow a written rule completely. Some need judgement inside a signed limit. Some carry a cost of error so high, or an action so final, that a person must decide. A few have never been agreed at all.

Herbert Simon drew the first line in The New Science of Management Decision (1960), separating programmed decisions from non-programmed ones. The four classes, mechanical, bounded, consequential and contested, are native to embedding practice. They live in the decision register.

The Payment run holds all four. Selecting the invoices due for this run is mechanical. Choosing the payment method within policy on discounts and cash is bounded. Releasing the proposal to the bank is consequential, because cash once sent rarely comes back. Deciding which suppliers to pay early when cash is tight is contested, because treasury and procurement pull different ways.

Use it at Sort, writing each decision as a question with its class and reason. A decision is mechanical only if its rule is written and uncontradicted. The common misreading is to class decisions by comfort, so that everything turns consequential because nobody wants to let go.



Like medicines: *some sold off the shelf, some only up to a set dose, some only on a doctor's signature, and a few the doctors have never agreed on.*

The Contested Decision

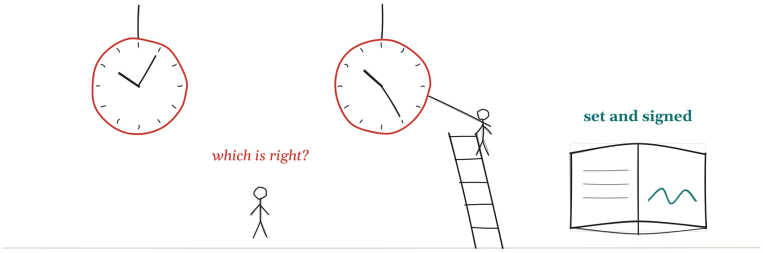
Some questions have no answer inside the company's rules, because the company never chose one. Two senior people would answer them differently, and both would cite policy. Better analysis cannot settle it. Someone with authority has to decide it and write it down.

Kurt Gödel showed in 1931 that any consistent formal system rich enough for arithmetic contains true statements it cannot prove. The lesson carries over by

analogy: some questions can only be settled from outside the system. The instrument, the contested-decision log, is native to embedding practice.

In Period close, an error found late forces a choice: book it now or next period. Controllers and finance chiefs often weigh that materiality call differently. At the manufacturer, the log held one entry, on disputed invoices from single-source suppliers. The chief financial officer and the chief operating officer settled it in thirty minutes.

Use it at Sort by asking, in the room, which decisions two senior people would answer differently. The lifecycle does not pass to Design while any entry stays open. The common misreading is that a capable executor can work out the answer, when it can only be given a rule the company already has.



Like two station clocks that disagree: *a sharper porter cannot say which is right, so the stationmaster sets one and signs the log.*

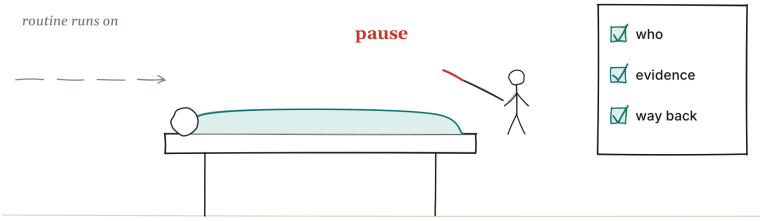
The Junction

An executor can run routine cases on its own, but some decisions must come back to a person. Where they return depends on what an error would cost and whether it can be undone, not on how comfortable anyone feels.

Its ancestor is the cord Toyota workers could pull to stop the line, described by Taiichi Ohno in Toyota Production System (1978). Thomas Sheridan and William Verplank set out graded levels of shared human and machine control in the same year. The junction itself is native to embedding practice and lives in the junction register.

In Candidate screening, law places a junction as well as cost. GDPR Article 22 restricts decisions with legal or similarly significant effects that rest solely on automated processing, as read in 2026. A rejection is such a decision. So a recruiter takes each rejection, and the executor lays the evidence in front of them.

Use it at Design. Each junction names who decides, what evidence they see and how an error is reversed. The common misreading is that any human approval is a junction, when an approval with no designed evidence, no named holder and no way back is a rubber stamp.



Like the pause before the first cut in surgery: *routine work runs on, but at the step that cannot be undone a named person checks the evidence and decides.*

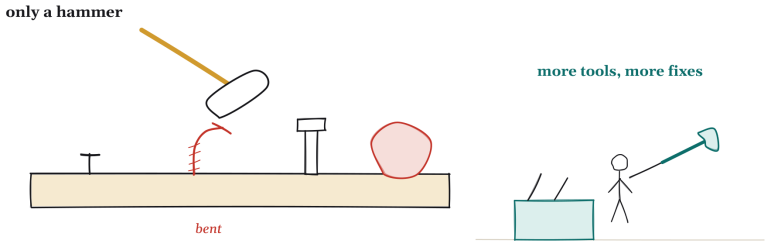
Requisite Variety

A controller can only handle as many kinds of trouble as it has kinds of response. Five rules cannot meet fifty kinds of case. The missing responses must come from somewhere, or the cases they would have met will fail.

W. Ross Ashby stated this as the law of requisite variety in An Introduction to Cybernetics (1956). His own short form was that only variety can destroy variety. In a loop, the variety comes from three places: written rules, a learned model and people.

Warranty claims make the law easy to see. Flat-rate labour tables and part lists handle the body of claims well. They have no response to a pattern that looks like fraud, or to a failure nobody has seen before. A learned model adds variety for messy claim descriptions. People at the edges add judgement for the cases nobody anticipated.

Use it at Design to check that each part of the distribution meets enough variety: rules for the body, the model for messy inputs, people for the tail. The common misreading is that a capable model removes the need for people, when the unforeseen tail still outruns it.



Like a toolbox holding only a hammer: *the nails go in, but screws and bolts need other tools, and the strange fitting needs a craftsman.*

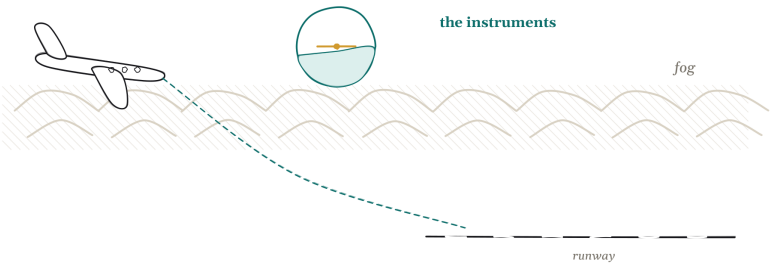
The Good Regulator

You cannot steer well what you do not represent. A controller with no picture of the system it controls succeeds only by luck. The better the picture, the better the control.

Roger Conant and W. Ross Ashby proved this in 1970, in a paper whose title states the result: "Every Good Regulator of a System Must Be a Model of That System". In embedding practice, the model is the loop's record set. It includes the map, the canvas, the registers and the standard.

Inventory replenishment is a regulator in plain sight. The reorder point is computed from a forecast, a lead time and a service level. Suppose the lead time on record is two weeks and suppliers now take five. Every order will then be wrong, however quickly it is placed.

Use it at Core and in the operating review. When outcomes slip, check the model before blaming the executor, and update the registers when the work has moved. The common misreading is that the registers and the standard are paperwork, when they are the regulator itself.



Like a pilot landing in fog: *the instruments are a model of the plane and the ground, and without a true model every landing is luck.*

The Closed Loop

A system that acts without checking its results can only hope. A system that acts, measures the result, compares it with a goal and adjusts can correct itself. The difference grows as the executor takes more of the decisions.

Walter Shewhart applied the cycle to factory quality in Economic Control of Quality of Manufactured Product (1931). Norbert Wiener turned feedback into a general science of control in Cybernetics (1948).

Support ticket resolution shows how a loop can look closed while running open. A ticket marked resolved is not proof of a customer helped. The standard therefore counts reopened tickets and repeat contacts. That way, a fast answer that failed comes back to the grader instead of vanishing.

Use it from Shadow onward. Grade every outcome, review results on a fixed rhythm, and turn findings into changed rules, limits or context. A loop without grading, review or a named owner is open. The common misreading is that a dashboard closes a loop, when nothing closes until someone adjusts.



Like watering a plant: *a timer sprinkler waters even in the rain, while a gardener feels the soil, compares, and waters only when it is dry.*

Cost per Accepted Outcome

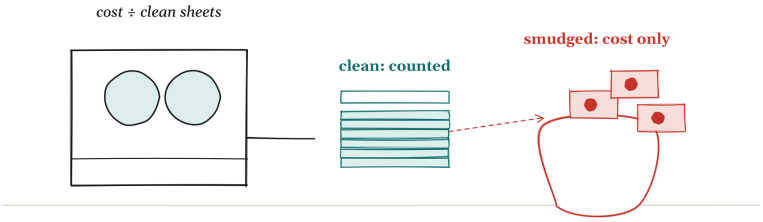
Cheap work that must be redone is not cheap. The honest unit cost divides everything a loop spends by the results that were actually good. Results that fail the standard add to the cost and nothing to the count.

Philip Crosby made the cost of poor quality visible to executives in Quality Is Free (1979). The measure itself is native to embedding practice. It lives on the economics sheet, signed by finance rather than by the programme.

At the manufacturer, the baseline was €11.80 per invoice. After embedding, it was €4.10 per accepted invoice, measured over eight weeks that included a

quarter-end. Split by part, body invoices cost €2.30 each and tail invoices €14.60. The average alone would have hidden that the tail barely moved.

Use it at Own. Take the baseline before the executor starts, and report the figure separately for body and tail. The common misreading is to divide by cases processed, which makes a loop that moves fast and creates rework look cheap.



Like a print shop that prices each clean sheet: *the ink, paper and time are divided only by the sheets that came out right, never by the smudged ones.*

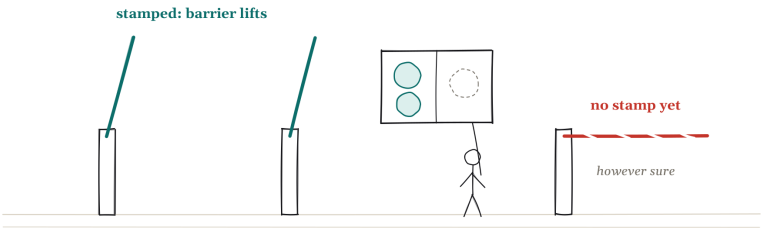
Scale Follows Signatures

Confidence is a poor reason to give a system more scope. A pilot that impressed a steering committee proves little about the cases it never met. A loop should widen only when named people have accepted, on record, the documents that say what it may do.

Robert Cooper's Stage-Gate process, set out in Business Horizons (1990), opened each stage of product development only on a gate decision. In embedding practice, the principle lives in the Scale Gates and the gate review record.

The five gates run from Mapped to Core. At G1, Shadow, the outcome, compensations, rules, context and standard must be signed. At G2, Bounded, the decisions, junctions and exceptions are signed too. For Candidate screening, a high-risk use under Annex III of the EU AI Act as read in 2026, those signatures also become evidence for a regulator.

Use it at every gate, and above all at Scale: no wider scope without the signatures the gate requires. A signature needs a name, a date and a record that outlives the meeting. The common misreading is that signatures slow the work, when they let the next widening happen without reopening every argument.



Like crossing borders: *each barrier lifts for the stamp in the passport, never for how sure the traveller feels, so the journey widens one stamp at a time.*

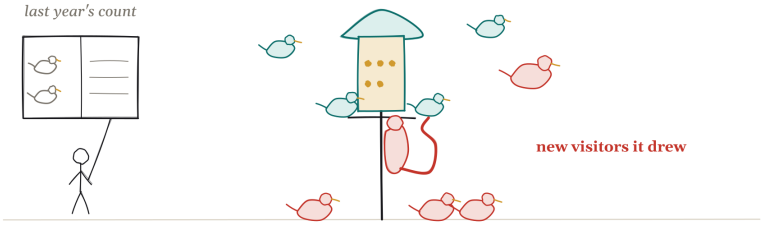
Reflexivity

A loop that acts on the world changes the cases it will meet next. Chase customers sooner and they change how they pay. Answer applicants faster and different people apply. The reading that justified a design goes stale precisely because the design worked.

George Soros called this reflexivity in The Alchemy of Finance (1987): participants′ views change the reality those views describe. Robert K. Merton had described a close cousin, the self-fulfilling prophecy, in The Antioch Review (1948).

Collections and cash application is reflexive by nature. Earlier reminders and better matching change payment habits. The late payers who remain are fewer and harder than before. A body-and-tail split drawn from last year′s ledger no longer fits this year′s cases.

Use it at Scale and in the map review. Re-read the distribution after a loop has run for a while, before widening its scope or choosing the next loop. The common misreading is that a stable result proves a stable world, when the loop may be holding steady on cases it has itself reshaped.



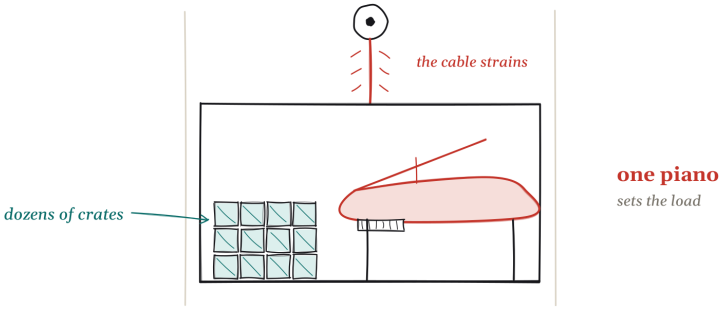
Like a garden bird feeder: *once it is filled, new kinds of birds and a squirrel arrive, so last year′s count of the garden no longer describes it.*

XXXIII

Heavy-Tailed Loops

Some loops look busy everywhere and matter in only a few places. Most of their cases are small and routine, and a handful carry most of the value, the loss or the effort. You can recognise one by sorting a quarter of cases by value and finding that the top few percent hold half the money. Another sign is that the team can name its difficult accounts without looking them up. Typical examples among the reference loops are Supplier invoice approval, Customer credit check, Collections and cash application, Supplier onboarding, Warranty claims and Inventory replenishment.

These loops punish designs that treat every case alike. An executor tuned on the average will do well on the many small cases and badly on the few that decide the result. The models in this chapter help a business engineer see where the weight sits, name more than one kind of tail, and set limits that hold when the large case finally arrives.



Like a goods lift carrying crates and one grand piano: *the many crates are routine, but the piano alone decides whether the cable holds.*

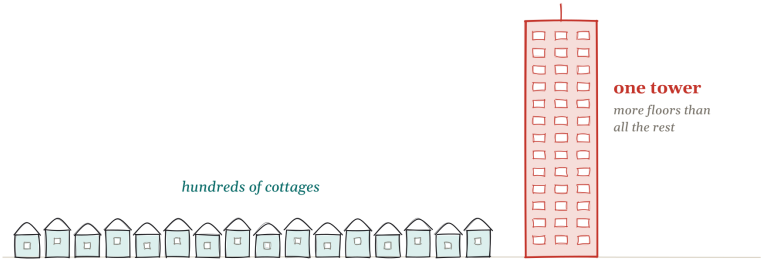
Power Laws and Extremistan

In some kinds of work, one case can be larger than thousands of others put together. Adding more cases does not smooth the picture.

Vilfredo Pareto observed this shape in the distribution of income in 1896. Benoit Mandelbrot showed in 1963 that cotton prices moved with the same wild jumps. Nassim Nicholas Taleb, in The Black Swan (2007), named the two worlds Mediocristan, where no single case matters much, and Extremistan, where one case can matter most.

Supplier invoice approval sits in Extremistan by value. At the manufacturer, 81% of invoices were under €5,000 and carried 14% of spend. The 3% over €50,000 carried 46%. A clerk's day is spent in the body, while the money is in the tail.

Use the model at Read, before anything is designed. Sort a few months of cases by value and compare each band's share of cases with its share of value. If a small band carries a large share, plan the loop as two loops: a body the executor can take and a tail that people must hold. The common misreading is that a power law is rare and exotic, when most money-carrying loops have one.



Like a town of cottages and one tower: *add a hundred more cottages and little changes, but the one tower holds more floor space than all of them.*

The Matthew Effect

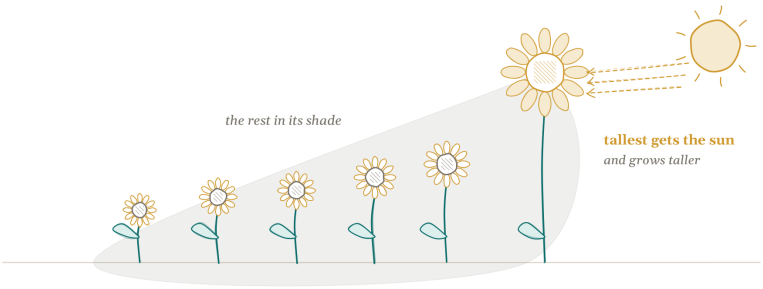
Advantage tends to feed on itself. The customer who buys more gets better terms and so buys more again. Over time, the gap between the largest cases and the rest widens rather than closes.

The sociologist Robert K. Merton named this the Matthew effect in a 1968 paper in Science, after the verse about those who have being given more. He showed how already famous scientists received more credit for the same work. Albert-László Barabási and Réka Albert showed in 1999 that this preferential attachment produces power laws in networks.

In Collections and cash application, a few large customers become more important each year. Their balances grow, their remittances become more complex, and their disputes carry more value. The same happens with strategic suppliers in Supplier onboarding. Each new contract deepens a relationship that was already the largest.

Use the model when you Map a loop, by asking whether the cases come from relationships that compound. If they do, expect the tail to get heavier over time and plan to re-read the distribution at each review. The common misreading is

that today's concentration is a fixed fact, when it is usually a trend still running.



Like sunflowers in a crowded bed: *the tallest catches the most sun, grows taller still, and leaves the rest a little deeper in its shade each week.*

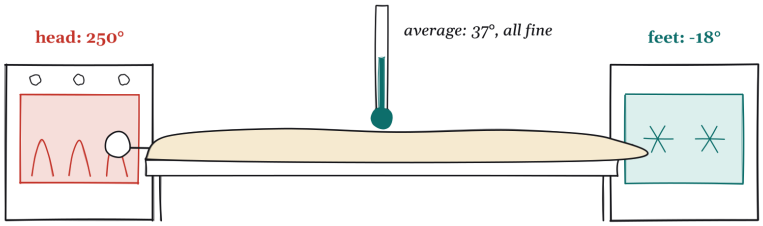
Averages Mislead

A loop can be fine on average and failing where it matters. The average invoice may be small and simple, while the invoices that move the quarter are large and messy. A plan built for the average case fits almost no real case.

Sam Savage called this the flaw of averages in his 2009 book of that name. He showed that plans built on average inputs go wrong when the inputs vary. In heavy-tailed data the error grows worse, because the average is pulled by a few giants and describes neither the body nor the tail.

The manufacturer saw it in shadow running. Overall agreement between the executor and the clerks looked excellent. Split by band, agreement was 97.8% on invoices under €5,000 and 81% on invoices over €50,000. The headline figure had been carried by the many small cases.

Use the model whenever a number goes into the standard or the economics sheet. Report every measure for the body and the tail separately, and refuse a single blended figure for a heavy-tailed loop. The common misreading is that a large sample makes the average safe, when in Extremistan it can make it less representative.



Like lying with your head in an oven and your feet in a freezer: *on average you are at body temperature, and no part of you is comfortable.*

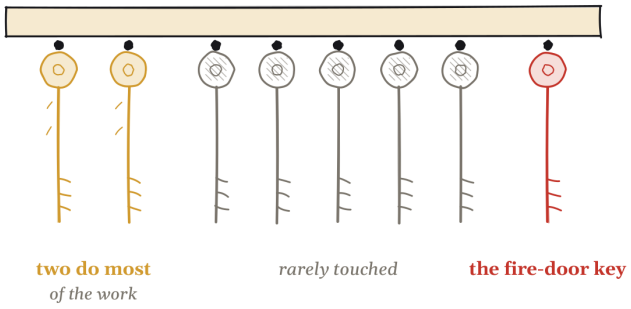
The Pareto Principle and Its Misuse

A small share of causes often produces a large share of effects. A few suppliers account for most spend. A few failure modes cause most returns. Finding those few is often the quickest route to a result.

Joseph Juran turned Pareto's income observation into a management rule in his Quality Control Handbook of 1951. He called it the vital few and the trivial many, and later regretted the word trivial. The figure of 80 and 20 became the popular shorthand, though Juran never claimed it as a law.

In Warranty claims, a few defect families usually drive most of the cost. That makes it tempting to automate the many small claims and leave the rest alone. Yet the trivial many still need handling, and a cheap claim can carry a safety signal.

Use the principle at Sort to choose where attention goes first, and then check the actual split in your own data. The numbers rarely come out at 80 and 20, and they need not add to one hundred. The common misreading is that the vital few deserve all the care and the many deserve none, when the many still need rules, an owner and a standard.



Like a key rack by the door: *two keys do most of the work, yet the dusty fire-door key is the one you need on the worst night.*

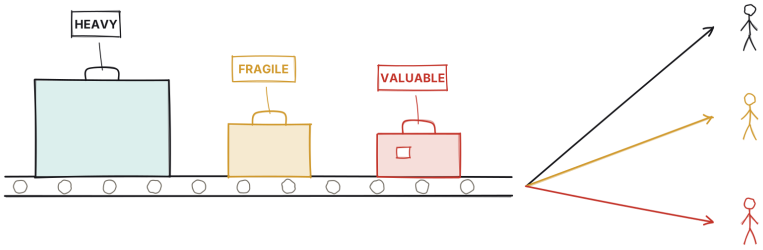
Segmenting by Value, Difficulty and Risk

Cases differ in more than size. A case can be large and easy, small and hard, or small and dangerous. Handling them well means sorting on each of these separately before deciding what goes where.

H. Ford Dickie described ABC analysis at General Electric in 1951, grading inventory by annual value so that the costly items got the closest control. Peter Kraljic added a second axis in Harvard Business Review in 1983, sorting purchases by profit impact and by supply risk. Embedding practice adds difficulty as a third dimension.

In Customer credit check, a large order from a long-standing customer is high in value and low in difficulty. A small first order from a new company is low in value and high in risk. A single value threshold would wave the second through. Inventory replenishment shows the same split between fast movers and rare, critical spares.

Use the three dimensions at Read and carry them into Bound. The executor's limits are then written on several conditions at once, as the manufacturer did with value, tolerance and supplier history. The common misreading is that segmenting by value alone is enough, when risk often hides in the small cases.



Like baggage tags at check-in: *a case can be heavy, fragile or valuable, and each tag sends it down a different chute to a different handler.*

Concentration Risk

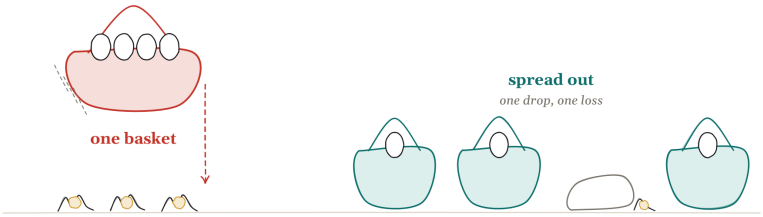
When much depends on a few, the loss of one of them is a large loss. One customer, one supplier or one claim can carry enough exposure to hurt the whole business. That is a fact about the portfolio, not about any single case.

Albert Hirschman measured concentration in foreign trade in 1945, and Orris Herfindahl applied a similar index to the steel industry in 1950. Banking regulators turned the idea into limits. The Basel Committee's large exposures

framework of 2014 caps a bank's exposure to one counterparty, as read in 2026.

At the manufacturer, the contested-decision log held one entry, on disputed invoices from single-source suppliers. Holding payment protected cash. It also risked a plant stopping for want of a part. That dispute existed because one supplier carried a whole production line.

Use the model at Bound and Own. Write portfolio limits into the decision register, not just case limits, so an executor cannot approve many medium cases that add up to one large exposure. Name the owner of each concentration in the charter. The common misreading is that concentration risk is a treasury matter, when operating loops create it every day.



Like carrying every egg in one basket: *one stumble breaks the lot, while eggs spread across several baskets lose only one when a basket drops.*

The Tail Has Many Definitions

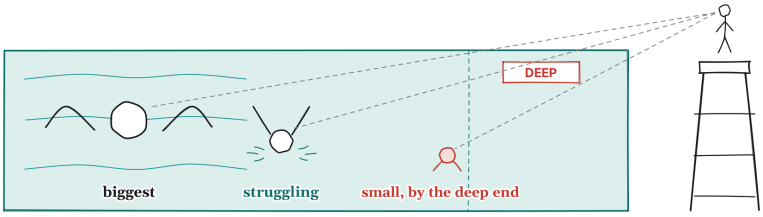
The tail is not one group of cases. The largest invoices are a tail by value, the oddest a tail by difficulty, and those that could hide fraud a tail by risk. They overlap less than people expect.

This model comes from embedding practice and lives in the reading of the distribution at Read. The reading names the body and the tail on each dimension that matters. It also names cases that are new, since a case the loop has never seen belongs to the tail whatever its size.

Supplier invoice approval shows the gap clearly. The large invoice from a supplier paid for years sits in the value tail and the difficulty body. A €900 first invoice with changed bank details sits in the value body and the risk tail. Each needs a different control.

Use the model when writing the junction register. Give each tail its own junction and its own holder, because whoever judges a large commercial dispute is rarely the right judge of a bank-detail change. Cases in two tails at

once need the most attention. The common misreading is that the tail means the big cases, when the most dangerous cases are often small.



Like a lifeguard scanning a pool: *the biggest swimmer, the struggling one and the small child by the deep end each need watching for a different reason.*

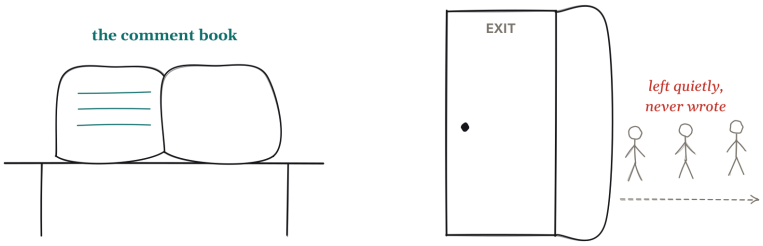
The Unseen Case

The cases that matter most can be the ones missing from the data. A sample built from cases that were easy to find leaves out those that were hard. A loop can look safe because its failures never reached the record.

Abraham Wald made the point in 1943 while studying damage on bombers returning from missions. The holes showed where a plane could be hit and survive, so the armour belonged where the returning planes had none. Frédéric Bastiat had argued in 1850 that good judgement weighs what is not seen as well as what is.

In Warranty claims, the cases on file are the ones customers chose to submit. Failures customers absorbed, or claims abandoned halfway, never appear. A model trained on filed claims learns little about them. The same applies to fraud in Supplier invoice approval, where the record holds the fraud that was caught.

Use the model at Shadow, by testing on the live stream of cases rather than a prepared set. Ask, for each data source, which cases could never have reached it. Record the gaps in the standard. The common misreading is that a large dataset covers the tail, when collection itself can remove it.



Like a restaurant's comment book: *it holds the views of diners who stayed to write, not of those who left quietly and never came back.*

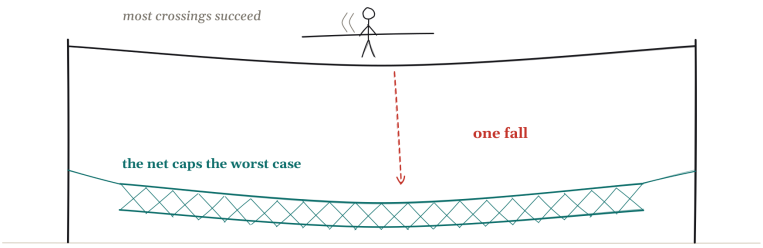
Expected Value and the Worst Case

Two decisions with the same average payoff can carry very different dangers. One loses a little, often. The other rarely loses, but its losses are ruinous. In heavy-tailed loops, the average cannot tell them apart.

Daniel Bernoulli made the point in 1738 with the St Petersburg paradox, a game whose average payoff is infinite but which no sensible person would pay much to play. Philippe Artzner and colleagues set out in 1999 which risk measures behave sensibly. Expected shortfall, the average loss in the worst cases, became a standard alternative to the plain average in banking.

In Customer credit check, approving a slightly risky customer is usually profitable on average. Yet one large default can wipe out a year of margin on small accounts. The average says approve. The worst case says cap the limit and route it to a person.

Use both numbers on the economics sheet. Put the expected cost per accepted outcome beside the largest loss a single error could cause, and set the executor's limit so that worst loss is one the business can absorb. The common misreading is that a positive average makes a decision safe, when ruin can sit inside a good average.



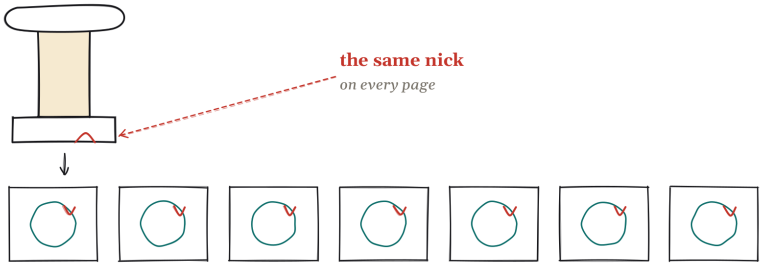
Like a tightrope walk: *most crossings succeed, so the average looks fine, but a single fall is final unless a net caps how far you can drop.*

XXXIV

High-Volume Routine Loops

Some loops are defined by repetition. Thousands of cases arrive each month, and most of them look alike: an invoice that matches its order, a payment that matches its invoice, a password reset, a standard order from a known customer. You can recognise these loops by their queues, by teams measured on cases per hour, and by a month-end backlog that everyone treats as weather. Typical reference loops are Supplier invoice approval, Collections and cash application, Billing and invoicing, Order booking, Support ticket resolution and Employee expense claims.

In these loops the large body of similar cases is where an executor earns its keep, and where most of the money moves. The danger is not that a case is hard. It is that small errors repeat thousands of times, that waiting hides inside volume, and that the measures meant to steer the loop start to steer it wrong. The models below are about flow, variation and measurement at scale.



Like a rubber stamp with a nick in its face: *every page carries the same small flaw, because in routine work one error repeats thousands of times.*

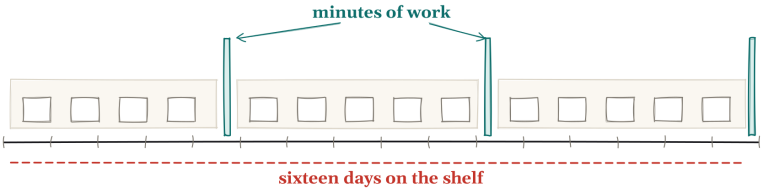
Flow Efficiency

Most of the time a routine case spends in a loop, nobody is working on it. It sits in an inbox, waits for a batch, or waits for someone to come back from a meeting. Divide the minutes of real work by the days from arrival to outcome, and the share is usually tiny.

The measure comes from lean practice. Niklas Modig and Pär Åhlström gave it its current name and its central place in *This Is Lean* (2012). Earlier, George Stalk and Thomas Hout had shown in *Competing Against Time* (1990) how little of elapsed time adds value.

In high-volume loops the waits dominate. At the manufacturer, twenty-three minutes of work on a supplier invoice took sixteen days, a flow efficiency well under 1%. The minutes were spread across scanning, coding, matching and a Friday approval round. The days were spent between them.

Measure flow efficiency at Map, from the walk-through and the baseline, before anyone builds anything. It tells you whether the prize is in the steps or in the waits between them. The common misreading is that a faster executor improves flow, when it only shortens the minutes and leaves the days untouched.



Like a parcel in transit: *handled for minutes and parked on shelves for days, so a faster sorting machine barely changes when it arrives.*

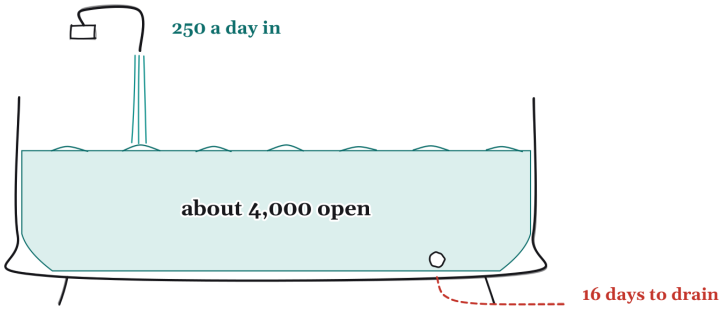
Little's Law

The number of cases sitting in a loop at any moment equals the rate at which they arrive multiplied by the time each one spends inside. If cases take twice as long, twice as many pile up.

John Little proved the relationship for queues in general in "A Proof for the Queuing Formula: L = λW" (1961). It holds for almost any stable system, whatever the order in which cases are served.

It turns a vague backlog into arithmetic. The manufacturer receives about 7,400 invoices a month, roughly 250 a day. At sixteen days each, around 4,000 invoices are open at any moment. Each open invoice is a supplier who may call, a discount that may lapse and a cost that sits unrecorded at month-end. Cut the time to four days and the open stock falls to about a thousand, with no change in volume.

Use it in the economics sheet and the baseline to price waiting, and in Design to size the exception queue that the tail will create. The common misreading is that a backlog proves a team is short of people, when it often proves only that cases wait too long between steps.



Like a bath with the tap running: *how full it gets is the flow in times how long each litre stays, so clear the drain before buying a bigger tub.*

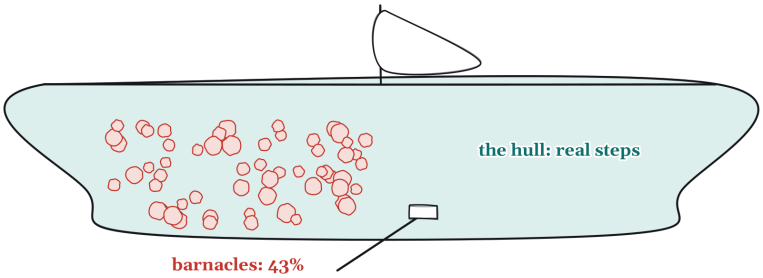
The Strip Ratio

Count the steps on a process map, then count how many exist only to work around something that no longer binds: an overnight batch, a paper signature, a second check added after an old error. The share you can strike tells you what kind of job lies ahead.

This model is native to embedding practice. It lives in the strip register, signed at Step 3, Strip. It builds on the lean idea of separating work that creates value from work that does not, but it asks a narrower question about each step: what constraint does this step compensate for, and does it still apply?

Routine loops gather compensations faster than most, because every past error at volume left a check behind. The manufacturer's invoice loop had a strip ratio of 43%. Printing and scanning PDFs were among the struck steps.

Use the ratio to decide the shape of the work. Under about a fifth, let an executor run what remains. Over about a third, treat the project as a redesign and strip before you build. The common misreading is that a high ratio blames the people, when the compensations were usually sensible answers to constraints that later disappeared.



Like scraping barnacles off a hull: *if a little comes away, sail on; if the crust is two-fifths of the boat, it needs the dry dock.*

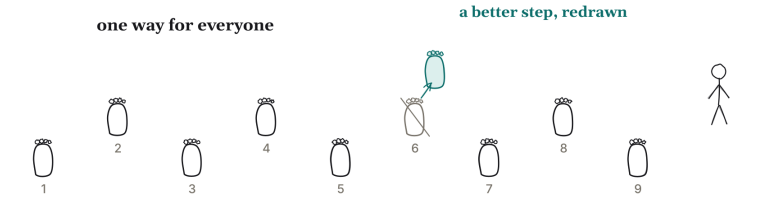
Standard Work

When thousands of similar cases pass through a loop, they should be handled the same way each time. Write down the current best way to handle the routine case, then make sure everyone uses it. Improve it on purpose, not by drift.

The idea comes from the Toyota Production System. Taiichi Ohno described it in *Toyota Production System: Beyond Large-Scale Production* (1978, English edition 1988). Standard work was never meant to freeze a method. It was meant to give improvement a fixed baseline to improve from.

In Order booking, most orders from known customers at contract prices need no human touch. Yet in many companies two clerks still book the same order differently, with different checks and different notes. An executor trained on that history learns both habits and the inconsistency between them.

At Design, the standard for the routine case should describe one accepted outcome and one way of grading it. When an exception reveals a better method, change the standard and the rule register together, through the operating review. The common misreading is that standard work removes judgement, when its real job is to free judgement for the cases that need it.



Like dance steps chalked on the studio floor: *everyone takes the routine case the same way, and when a better step is found, the chalk is redrawn for all.*

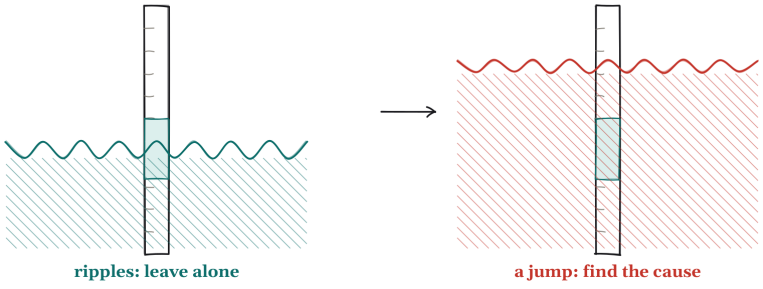
Common and Special Causes

Every high-volume loop wobbles. Some days more cases fail, some days fewer, for reasons nobody can name. That ordinary noise belongs to the system. Occasionally something new happens and moves the numbers for a reason you can find. The two call for opposite responses.

Walter Shewhart drew the distinction in *Economic Control of Quality of Manufactured Product* (1931) and built the control chart to separate them. W. Edwards Deming named them common and special causes and spread the idea in *Out of the Crisis* (1986).

In Collections and cash application, the daily share of unmatched payments moves up and down by a few points with no single cause. Then a large customer changes its remittance format, and unmatched cash jumps and stays high. Chasing the first wobble wastes effort. Ignoring the second leaves cash unapplied for weeks.

Plot the share of outcomes meeting the standard as a control chart in the operating review. React to points outside the limits by finding the cause, and improve the system to narrow the limits themselves. The common misreading is that every bad day deserves an explanation, which leads teams to tamper with a stable loop and make it worse.



Like a river gauge post: *daily ripples inside the band are just the river, but a rise that jumps the band and stays has a cause upstream.*

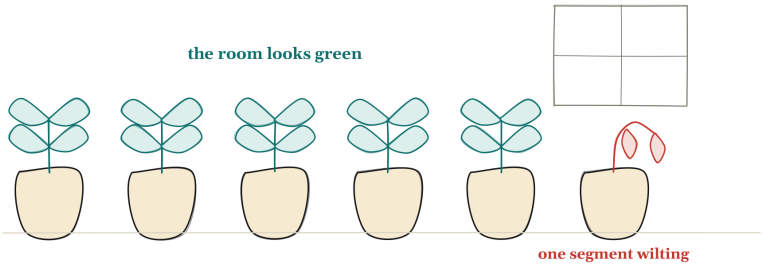
Drift

A loop can get slowly worse without anything breaking. Suppliers change their layouts, customers change how they write, a model is updated, and the people who handle exceptions change. No alarm sounds. The share of good outcomes simply slides.

This is native to embedding practice, and it is watched through the standard and the operating review. It borrows from machine learning research on concept drift, described by Gerhard Widmer and Miroslav Kubat in "Learning in the Presence of Concept Drift and Hidden Contexts" (1996).

Volume makes drift both dangerous and visible. In Billing and invoicing, national e-invoicing mandates change the formats and clearance channels invoices must pass. Germany's requirement to receive B2B e-invoices from 1 January 2025 is one example, as read in 2026. An executor tuned to last year's formats can fail on one segment while the overall figure barely moves.

Catch drift in Core by grading every outcome, checking the grader with a weekly human sample, and reading body and tail separately each month. When drift appears, use the switch-off on the affected segment. The common misreading is that drift is a model problem, when it usually starts in the inputs or in the people around the executor.



Like a row of houseplants where one by the far window slowly wilts: *the room still looks green, and only checking each pot finds the dying one.*

The Ironies of Automation

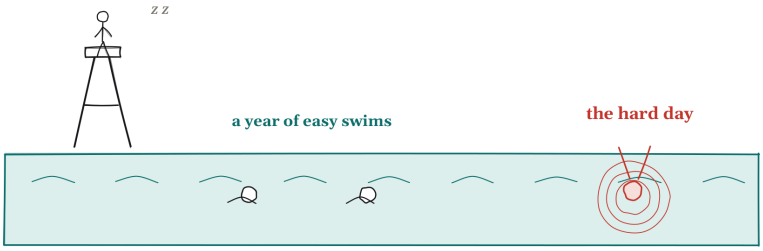
Automate the easy cases, and the people left behind get only the hard ones. They see fewer cases, practise less, and lose the feel for what normal looks like. Then, on the rare day the system fails, they are asked to take over at full volume with skills that have faded.

Lisanne Bainbridge set this out in "Ironies of Automation" (1983), written about process control and aircraft cockpits. Her point was that automation does not remove the human role. It changes it, and often makes it harder.

In Support ticket resolution, an assistant may absorb most routine contacts, as the published deployments in the reference loop suggest. The human agents then handle only escalations. When the assistant is switched off for an

incident, those agents must resolve password resets and billing questions at a pace they have not worked for months.

Plan for this at Own and test it with the switch-off. Rotate people through some routine cases, keep the switch-off drills real, and size the team for the day the executor stops. The common misreading is that the remaining human role is the easy part, when it is the hardest and least practised job in the loop.



Like a lifeguard at a pool where the swims are easy: *a year without a rescue, and the hard one arrives when the skills are least practised.*

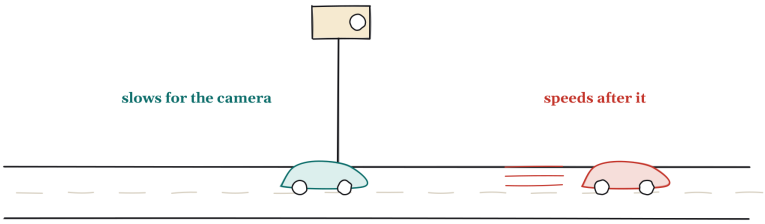
Goodhart's Law

A number that once told you how a loop was doing stops telling you once people are rewarded for moving it. They find ways to move the number that do not move the outcome. In routine work, with thousands of cases and simple counts, those ways are easy to find.

Charles Goodhart made the observation about monetary targets in "Problems of Monetary Management: The U.K. Experience" (1975). Marilyn Strathern gave it its familiar form in 1997: when a measure becomes a target, it ceases to be a good measure.

Support ticket resolution shows it plainly. Reward closure speed, and tickets are closed early and reopened later. Reward first-contact resolution, and hard tickets are passed on rather than solved. An executor optimised against the same count will learn the same tricks, only faster.

Define success in the standard as an accepted outcome, and include the customer's view through reopened cases, repeat contacts and complaints. Then report cost per accepted outcome in the economics sheet, not cost per case processed. The common misreading is that a better metric solves the problem, when any single number will bend once it carries a reward.



Like a speed camera: *drivers brake for the box and speed up after it, so once the reading is the target it stops describing the road.*

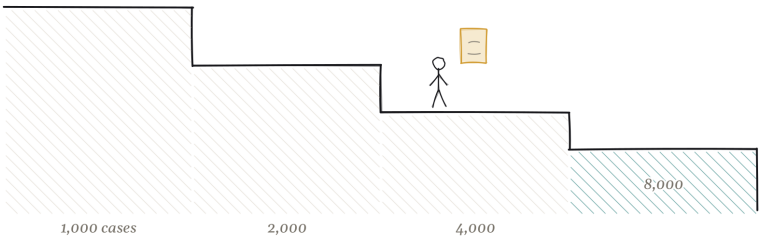
The Learning Curve

The more often a team does the same work, the cheaper each repetition becomes. The gain is steady and predictable. Every time cumulative output doubles, the cost of a unit falls by roughly the same share.

Theodore Wright measured it in aircraft assembly and published it as "Factors Affecting the Cost of Airplanes" (1936). The Boston Consulting Group later generalised it as the experience curve.

High-volume loops climb the curve quickly, because doublings arrive in weeks rather than years. At the manufacturer, each exception the controller resolved became a candidate rule or context fix. The exception queue shrank as volume accumulated. The same curve applies to the lifecycle itself: the first loop was the most expensive, and later loops ran faster and cheaper.

Plot cost per accepted outcome against cumulative cases in the economics sheet, and feed every resolved exception back to the rule register. At Scale, track the falling time and cost per loop as the company's real asset. The common misreading is that the curve comes free with volume, when it comes only from deliberately capturing what each case teaches.



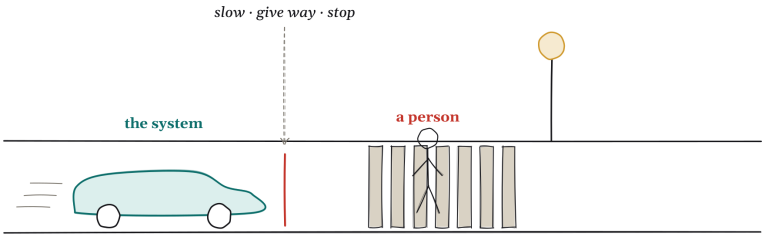
Like walking down a staircase where each step is a doubling of cases: *the cost drops a stair each time, provided every lesson is written down.*

XXXV

Loops That Decide About People

Some loops produce an outcome that lands on a named person. A candidate is shortlisted or rejected, a sole trader gets credit or a demand for prepayment, a pay change goes through, leave is granted, an employment ends. You can recognise these loops by asking who would complain if the decision were wrong. If the answer is an individual rather than a supplier's accounts team, the loop belongs here. In the reference set, the typical examples are Candidate screening, Customer credit check, Payroll changes, Leave and absence requests, HR helpdesk and Offboarding.

These loops are where an error costs most per case and where the law already says a good deal about how decisions must be made. The person affected cannot see the machinery and usually cannot walk away. So the models in this chapter are less about speed and more about fairness, evidence and a way back. They tell the business engineer which decisions the law has already classified, what each person is owed, and how a well-meant system can drift into treating groups unequally.



Like a zebra crossing: *wherever a machine's path meets people, it must slow down, give way and be able to stop, because what it hits is a person.*

High Risk by Law

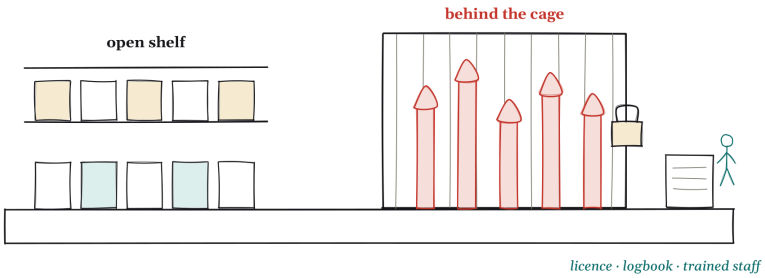
In most loops the company decides how careful to be. In some, the law has already decided. Certain uses of automated systems are named in advance as risky to people's rights, and they carry extra duties however well they work.

The clearest statement is the EU Artificial Intelligence Act, Regulation (EU) 2024/1689, as read in 2026. Its Annex III lists high-risk uses. Point 4 covers recruitment, selection and decisions affecting work relationships. Point 5(b)

covers scoring the creditworthiness of natural persons. Colorado's SB 24-205 takes a similar approach for consequential employment decisions.

The classification follows the use, not the vendor's label. In Candidate screening, a tool that filters applications is high-risk, so the deployer owes trained human oversight, logs and notice to workers. In Customer credit check, credit on companies falls outside point 5(b), but sole traders fall inside it.

Use it at Read and Sort. For each decision in the decision register, record whether a listed use applies and cite the rule in the rule register. The common misreading is that buying a compliant product makes the deployment compliant, when most of the deployer's duties stay with the deployer.



Like a fireworks counter: *the law, not the shopkeeper, lists which goods go behind the cage and need a licence, a logbook and trained staff.*

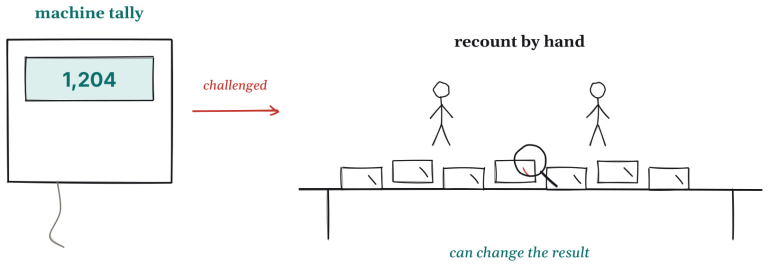
The Right to Human Review

When a decision with real effect on someone is made by a system, that person can ask for a human to look at it again. A person who can only confirm what the system said is not a review.

The right comes from Article 22 of the General Data Protection Regulation, Regulation (EU) 2016/679, as read in 2026. A decision based solely on automated processing, with legal or similarly significant effect, is barred unless an exception applies. Even then the person can obtain human intervention. The Court of Justice widened the reach in SCHUFA (C-634/21, 2023): a score that strongly drives a lender's decision counts as the decision.

In Offboarding, a dismissal sits under Article 22 and, in Germany, under a works-council hearing as well. In Candidate screening, a rejection has legal effect on the applicant. Both must reach a person with authority and the facts to reverse.

Use it at Design and Own. In the decision register, name the reviewing role, its authority and the evidence it sees. The common misreading is that a human somewhere in the loop satisfies the right, when what counts is a human who can actually decide the case.



Like a recount by hand: *when the machine's count is challenged, people with authority go back to the actual papers and can change the result.*

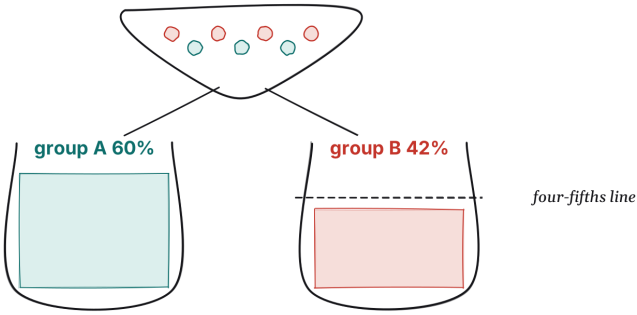
Adverse Impact and the Four-Fifths Rule

A selection step can be neutral in its wording and still pass one group far less often than another. The quickest test is to compare pass rates. Divide the lower group's rate by the higher group's rate and look at the ratio.

The Uniform Guidelines on Employee Selection Procedures of 1978, issued by the US Equal Employment Opportunity Commission and other agencies, made this a working rule. A selection rate for any group below four-fifths of the highest group's rate is generally taken as evidence of adverse impact. The idea rests on Griggs v. Duke Power Co. (1971), where neutral tests with unequal effects were held to need a business justification. New York City's Local Law 144 now requires published impact ratios for automated hiring tools, as read in 2026.

In Candidate screening, suppose a ranking agent advances 60 per cent of one group and 42 per cent of another. The ratio is 0.70, under the line.

Use it at Shadow and keep it in the standard, recomputed whenever the model or criteria change. The common misreading is that passing the ratio proves a tool fair, when it is only a screen that tells you where to look.



Like two jars filled by the same sorting machine: *if one group's jar fills to under four-fifths of the other's level, look hard at the machine.*

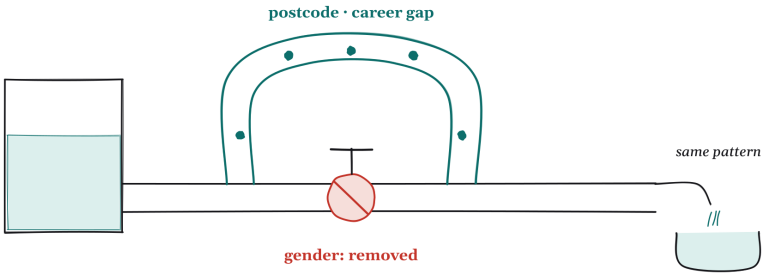
Proxy Discrimination

Deleting a sensitive field from the data does not remove what it predicts. Other fields often carry the same signal. A model trained to find patterns will find the substitute and use it, without anyone asking it to.

Solon Barocas and Andrew Selbst set out the mechanism in "Big Data's Disparate Impact" (2016). Anya Prince and Daniel Schwarcz named it proxy discrimination in 2020. In 2018 Reuters reported that Amazon had dropped an experimental recruiting tool that marked down CVs containing the word "women's".

In Candidate screening at the pump and valve maker, gender is never an input. But a gap in employment history, part-time spells or certain hobbies can stand in for it. In Customer credit check, a sole trader's postcode can stand in for ethnicity. The model never sees the protected trait, yet its outputs can still split along it.

Use it at Strip and Sort. For every input in a people decision, ask what it could be standing in for, and keep the answer in the rule register. Then test outcomes, not inputs, with the four-fifths ratio. The common misreading is that a model blind to a trait cannot discriminate on it.



Like shutting a valve while the water runs round the bypass: *remove the sensitive field and the pattern flows on through the inputs that stand in for it.*

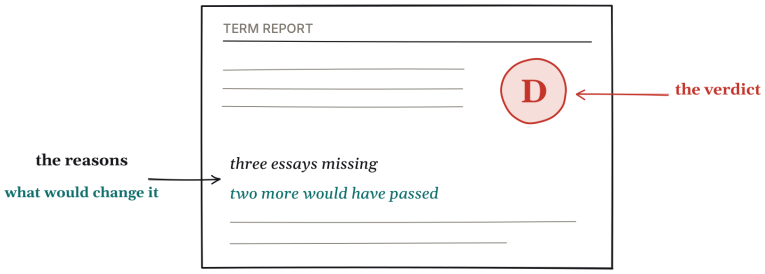
The Explanation Owed

A person turned down is owed more than the verdict. They are owed the main reasons, in words they can act on. A useful explanation says what drove the outcome and what would have changed it.

The US Equal Credit Opportunity Act of 1974 and its Regulation B require a creditor to state the principal reasons for adverse action. The GDPR adds a right to meaningful information about the logic of automated decisions, as read in 2026. Sandra Wachter, Brent Mittelstadt and Chris Russell proposed counterfactual explanations in 2018. These are statements of the smallest change that would have flipped the result.

In Customer credit check, Regulation B requires reasons for a declined small-business applicant, given or disclosed. Larger firms and trade credit can ask in writing within 60 days. The loop must keep the factors behind each decision so a credit manager can state them.

Use it at Design. Make "reasons stated" a required output in the decision register for every consequential people decision, and test it during Shadow. The common misreading is that an explanation means describing how the model works, when the person needs to know why their own case went as it did.



Like a school report: *a grade alone tells the pupil nothing; the teacher's comments say what drove it and what would have changed it.*

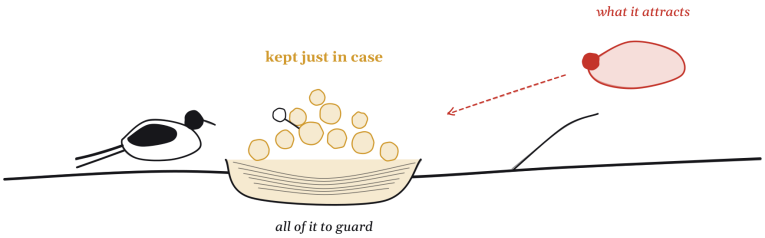
Data Minimisation

Collect only what the decision needs, and keep it only as long as the decision needs it. Every extra field is something to protect, to justify and, sooner or later, to explain.

The OECD's Guidelines on the Protection of Privacy of 1980 set a collection limitation principle. Article 5(1)(c) of the GDPR makes data minimisation binding, as read in 2026. Article 9 adds stricter conditions for special-category data such as health.

In Leave and absence requests, a medical certificate carries health data. An agent that approves routine holiday needs only the balance, the dates and the team calendar. It never needs the diagnosis behind a sick note. At the pump and valve maker, the HR helpdesk agent answers policy questions for all 4,200 people. It does not need their payslips to tell them how parental leave works.

Use it at Strip, when the context an executor sees is decided, and record it in the data protection impact assessment that Article 35 requires before go-live. The common misreading is that more data always makes a better model, when in people loops it mostly makes a larger liability.



Like a magpie's nest: *every shiny thing hoarded just in case becomes one more thing to guard, and the extras are what draw the thieves.*

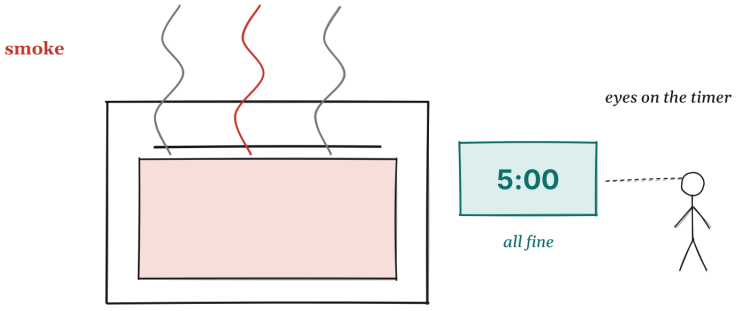
Automation Bias

People who check a system's output tend to accept it. The more reliable the system looks, the less closely they check. Over time the reviewer stops reviewing and starts approving.

Raja Parasuraman and Victor Riley described this misuse of automation in 1997. Linda Skitka, Kathleen Mosier and Mark Burdick measured it in 1999. Their participants followed wrong automated advice and missed events the aid did not flag. Article 14 of the EU AI Act requires that those who oversee a high-risk system stay aware of the tendency to over-rely on it, as read in 2026.

In Candidate screening, a recruiter shown a ranked shortlist with confidence scores will rarely reopen a rejected file. The human review exists on paper. In practice the agent decides. The same drift appears in Payroll changes when an approver signs off a batch the system has already marked as clean.

Use it at Shadow and Own. Watch the override rate. If reviewers almost never disagree, the system is near perfect or the review has gone hollow. Seed known errors to tell which. The common misreading is that adding a human reviewer removes the risk, when the reviewer's attention is the very thing the system erodes.



Like a cook who trusts the oven timer while smoke pours from the door: *the confident display wins over what the reviewer's own eyes would catch.*

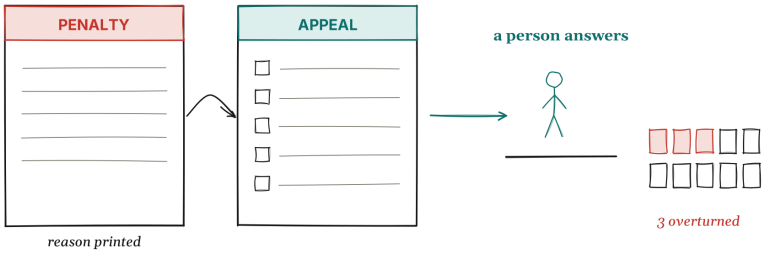
Contestability

A good people loop is built so that its decisions can be challenged. The person needs a channel, a reason to argue with, someone who must answer, and an outcome that can be reversed. The challenges, taken together, also tell the company where the loop is wrong.

Article 22(3) of the GDPR gives a right to contest automated decisions, as read in 2026. Marco Almada argued in 2019 that contestability should be designed into a system from the start, not bolted on. The Platform Work Directive, (EU) 2024/2831, requires written reasons and human review when a platform worker's account is suspended or ended.

In Leave and absence requests, refusing leave for business reasons is a contested decision. In Offboarding, a dismissal is the extreme case. Each needs a route for the employee to push back, reaching someone other than whoever decided.

Use it at Design and Own. Route every challenge into the contested-decision log, track how many succeed, and feed the pattern back into the rule register at each review. The common misreading is that a rising number of appeals means the system is failing, when a loop nobody can challenge simply hides its errors.



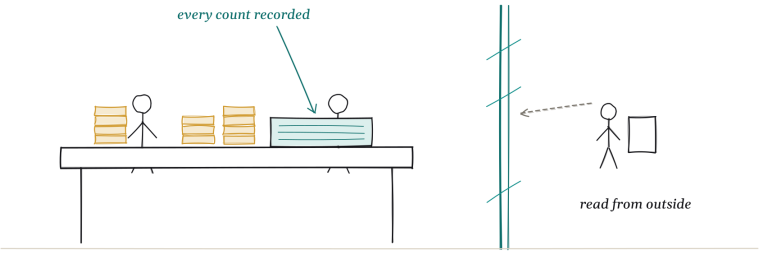
Like a parking ticket with an appeal form on the back: *the reason is printed, the appeal reaches a person, and the overturned tickets show where wardens err.*

XXXVI

Regulated and Audited Loops

Some loops are read twice. The people who run them read the records to get the work done, and then an auditor, a regulator or a finance controller reads the same records to decide whether the work can be trusted. These are the loops of record to report, payments and controls over financial reporting. You can recognise one by its sign-offs, its evidence requests and its calendar of quarter-ends. Typical examples among the reference loops are Journal entries, Account reconciliation, Period close, Audit requests, Payment run and Intercompany settlement.

In these loops a correct outcome is not enough. The company must also be able to show, months later and to a sceptical outsider, who did what, on what evidence, under which control. An executor that decides changes who does the work, but not who must answer for it. The models in this chapter are the ones auditors already think with, and the ones a business engineer needs in order to keep a loop auditable once it is embedded.



Like a casino counting room behind glass: *the staff count the cash, and an outsider can watch and later check every count they wrote down.*

Segregation of Duties

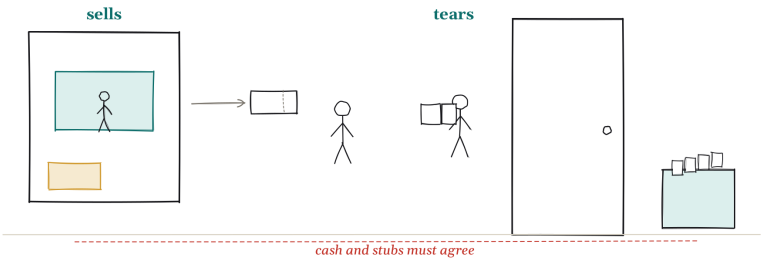
No single pair of hands should be able to start a transaction, approve it and hide it. When one person can create a supplier, enter that supplier's invoice and release the payment, an error or a fraud can travel from start to finish with nobody else seeing it. Split the steps, and a mistake must get past someone who had no part in making it.

Its modern form comes from the COSO *Internal Control: Integrated Framework* (1992, updated 2013). The Sarbanes-Oxley Act of 2002 made it a

matter of law for US-listed companies through management's annual report on internal control.

In a Payment run, the person who maintains supplier bank details must not be the person who releases the payment file. In Journal entries, a preparer cannot approve and post their own manual entry. At the manufacturer, the executor that codes invoices has no access to the supplier master.

Use it at Sort, tagging each rule in the rule register with the duties it separates, and at Design, when the executor's permissions are drawn. The common misreading is that an executor has no duties to segregate, when in fact it is the clerk with the widest reach of all.



Like a cinema where one person sells the tickets and another tears them at the door: *cash and stubs must agree, so neither can cheat alone.*

The Audit Trail

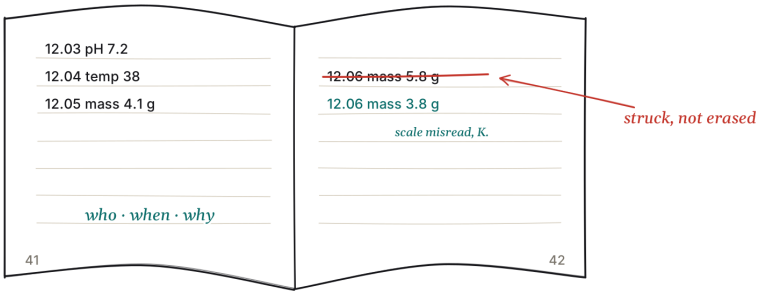
Every change to a record should leave a trace that says what changed, who or what changed it, when, and on what basis. A correction is a new entry that points to the old one, never an erasure.

Luca Pacioli's *Summa de arithmetica* (1494) described the journal and ledger that let a merchant follow any figure back to its source. Modern standards such as PCAOB AS 1105 *Audit Evidence* (2010), require auditors to test whether the information a company produces is complete and accurate. That test is only possible with a trail.

In Account reconciliation, the trail shows which items matched automatically, which a person explained, and who signed the balance. An executor adds a new kind of entry. Its trail must also record the model version, the rule it applied and the evidence it read, because an auditor will ask why a case was decided, not only that it was.

Write the trail requirement into the standard at Design, and test it during Shadow by asking someone outside the team to reconstruct ten decisions from

the log alone. The common misreading is that system logs are an audit trail, when most record activity rather than reasons.



Like a lab notebook with numbered pages in ink: *a mistake is struck through and corrected beside it, never erased, so anyone can see what changed and why.*

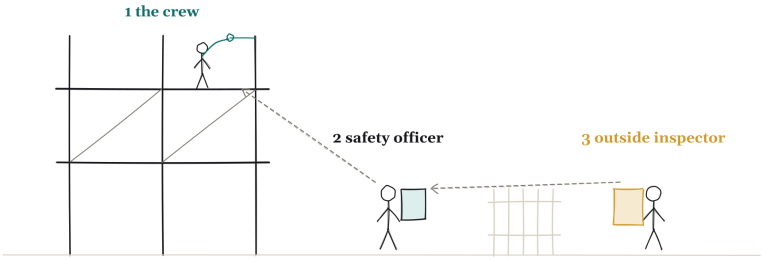
Three Lines of Defence

Risk is managed best when three groups each do a different job. The people who run the work own its risks and control them day to day. A second group sets the policies and watches whether they hold. A third group, independent of both, tells the board whether the whole arrangement works.

The model was set out by the Institute of Internal Auditors in its position paper *The Three Lines of Defense in Effective Risk Management and Control* (2013). It was revised as *The IIA's Three Lines Model* (2020), which stresses cooperation over separate walls.

In Period close, the accounting team runs the close checklist and signs each task. The financial controls function sets which reconciliations count as key controls and reviews exceptions. Internal audit tests a sample each year and reports to the audit committee. An embedded loop must fit all three lines, or one will reject it late.

At Own, name the first-line loop owner, the second-line control owner and the third-line contact in the loop charter. The common misreading is that the executor sits in the first line alone, when the second line must also be able to read its standard and its grading.



Like a building site: *the crew works safely, a site safety officer checks them daily, and an outside inspector checks whether that checking works.*

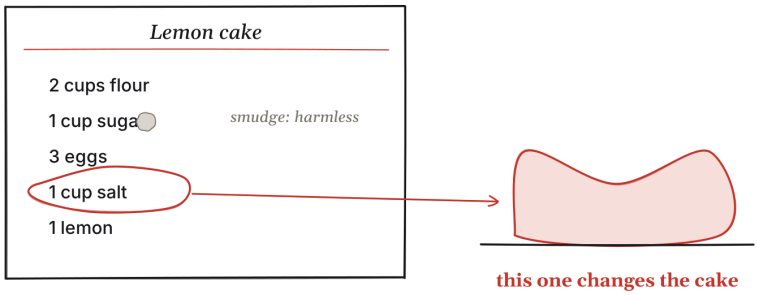
Materiality

An error matters when it is large enough, or of the right kind, to change the decision of someone who relies on the figures. A rounding difference of a few euros in a ledger of millions changes nothing. A small misstatement that turns a loss into a profit changes everything.

The accounting definition is in the IASB's *Definition of Material*, amendments to IAS 1 and IAS 8 (2018). In the United States, SEC Staff Accounting Bulletin No. 99 (1999) warned that a percentage rule of thumb is not enough, because a small amount can still be material for qualitative reasons.

In Period close, a materiality threshold decides which variances must be investigated and which late errors force the books to be reopened. In Intercompany settlement, a similar threshold decides when an unmatched difference may be written off. The threshold is set by people, not by the executor.

Record materiality thresholds as rules in the rule register at Sort, each with its source and owner, and use them to set the edges of bounded scope at Bound. The common misreading is that materiality is a single number, when qualitative factors such as fraud or a broken covenant override any size.



Like a typo on a recipe card: *a smudged letter changes nothing, but a cup of salt instead of a teaspoon ruins the cake, so only that error is fixed.*

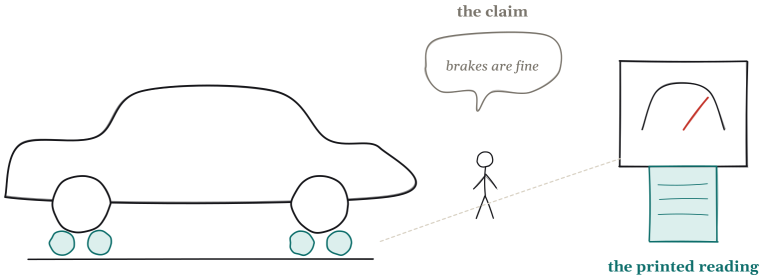
Evidence Over Assertion

A statement that something is true is a claim. Evidence is what lets someone else check the claim without trusting the person who made it. In audited work, "the reconciliation was reviewed" counts for little. A signed, dated review with the reconciling items attached counts for a great deal.

Audit standards call management's statements "assertions" and require auditors to gather evidence for each one. The international version is ISA 500 *Audit Evidence* (IAASB, 2009). The same stance runs through PCAOB AS 1105 (2010), which ranks evidence by its source and by how directly it was obtained.

In Audit requests, the auditor asks for evidence that a control operated, not for a description of the control. An executor can assemble that evidence faster than anyone, but its own summary is still an assertion. The packet must point to the source records, so the auditor can test them directly.

At Design, write the standard so that each criterion names the evidence that proves it, and make the junction register say what evidence the junction holder sees. The common misreading is that an executor's confident explanation is evidence, when it is only a well-written assertion.



Like a roadworthiness test: *the owner says the brakes are fine, but the garage runs the car on the rollers and prints a reading anyone can check.*

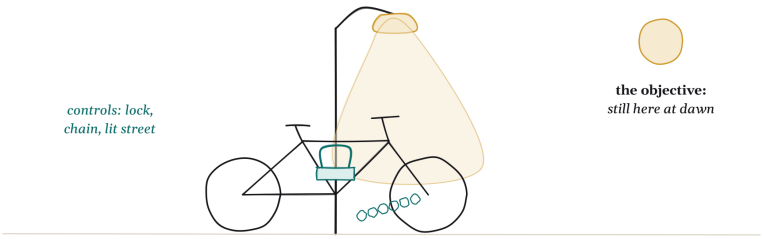
Control Objectives and Controls

Every control exists to make something true. The thing that must be true is the objective, for example "only valid suppliers are paid". The control is one way of achieving it, such as a call-back before bank details change. Several controls can serve one objective, and a control with no objective is only a habit.

The distinction was made explicit by ISACA's *Control Objectives for Information and Related Technologies*, known as COBIT (1996). Service-organisation reports under the AICPA's SSAE 18 (2017) still list control objectives first and the controls that meet them second.

In Supplier invoice approval, the objective "no invoice is paid twice" was once met by a clerk scanning for duplicates. Once the loop is embedded, the same objective can be met by a mechanical duplicate check on every invoice plus grading. The objective stays fixed while the control changes.

At Sort, list each control in the rule register beside the objective it serves, and agree with the control owner that objectives are signed and controls may be redesigned. The common misreading is to preserve every existing control unchanged, when auditors care whether the objective is still met.



Like keeping a bicycle overnight: *the goal is the bike still there at dawn, and the lock, the chain and the lit street are different ways to make sure.*

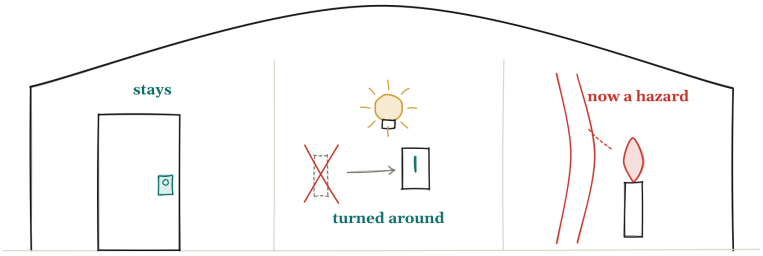
Controls That Survive and Controls That Invert

Controls written for people do not all carry over to an executor that decides. Some work exactly as before. Some keep their purpose but must be turned around to work. A few become harmful if they are left as they are.

This model comes from embedding practice and lives in the junction register and the standard. It is defined in the Control and Acceptance area of this book. It applies the previous model directly: because the objective is fixed, the control is free to change.

A call-back on changed bank details survives unchanged. Segregation of duties inverts. It becomes a separation between the executor and the standard it is graded against, with junctions at the consequential points. A four-eyes review becomes a grader plus a junction, and sampling becomes grading every outcome. "Review everything" becomes dangerous. Applied to thousands of decisions a day in Account reconciliation, it guarantees that reviewers stop reading.

At Design, sort every existing control into one of the three groups, and take the sorted list to the control owner before Bound. The common misreading is that a control which inverts has been weakened, when it usually covers more cases than the original did.



Like a house moving from candles to electric light: *the door lock stays, the nightly candle round becomes a switch, and candles left burning everywhere become a hazard.*

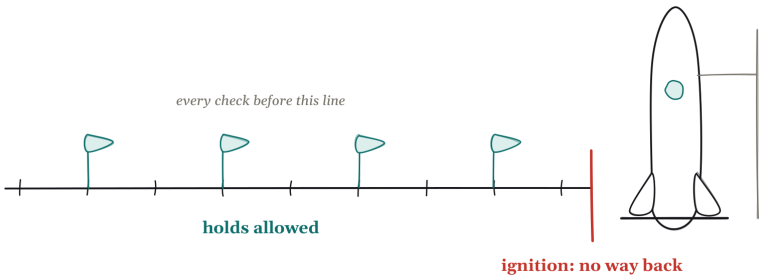
The Irreversible Action

Most mistakes in a loop can be undone. A wrong code can be reversed and a held invoice released. A few actions cannot be pulled back once taken, and they deserve a different kind of care. Every check that matters must sit before that point, because nothing placed after it can help.

This model comes from embedding practice and lives in the junction register, where junctions are placed by reversibility and cost. A familiar ancestor is Jeff Bezos′s distinction between one-way and two-way doors, in his letter to Amazon shareholders for 2015 (2016).

The Payment run is the clearest case. Once funds leave, recovery depends on the receiving bank and on the payee′s goodwill. Under the EU Instant Payments Regulation (2024), euro credit transfers settle in seconds, which removes the old overnight window to catch an error, as read in 2026. Locking a period at the end of Period close is another one-way door.

At Design, mark every irreversible action on the Work Loop Canvas and give each a junction with evidence and a named holder. The common misreading is to place junctions where people feel uneasy, when they belong where the action cannot be undone.



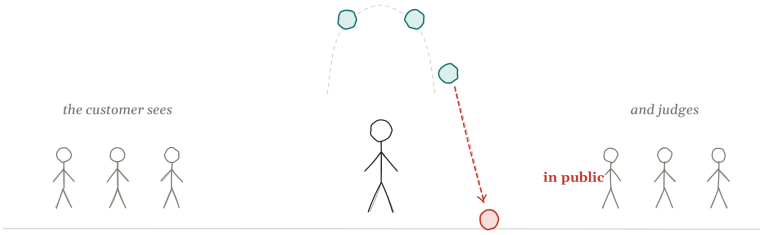
Like a rocket countdown: *holds are allowed at any point before ignition, so every check sits before that moment, because after it nothing comes back.*

XXXVII

Customer-Facing Loops

Some loops end in front of a customer. The outcome is an answer, a refund, a reply to a complaint, a call from a seller or a reminder to pay. You can recognise them by one test: the person who receives the result can see it, judge it and respond. Typical reference loops are Support ticket resolution, Returns and refunds, Complaint handling, Lead qualification, Collections and cash application and Warranty claims.

In a back-office loop, a weak outcome is caught by the grader or the review before anyone outside notices. Here the customer notices first, and the customer's reaction becomes part of the result. Words become commitments, a closed case can return as a second call, and a single badly handled minute can outweigh a fast process. The models in this chapter matter most here because they deal with that exposure: who else is grading, what the executor may say and promise, and which moments the customer will remember.



Like a street juggler: *practice happens out of sight, but every ball dropped in the act lands in front of the crowd who decide whether to stay.*

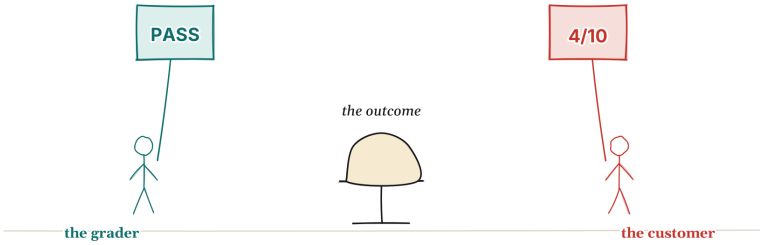
The Customer Grades Too

A company can write a careful standard, grade every outcome against it and still lose the customer. The customer applies a standard of their own. It rests on what they expected, and nobody inside the company wrote it down.

Service researchers made this gap explicit. A. Parasuraman, Valarie Zeithaml and Leonard Berry, in their 1985 model of service quality, defined quality as the distance between what a customer expects and what they perceive they received. Their later SERVQUAL instrument (1988) tried to measure that distance directly.

In Support ticket resolution the gap is easy to miss. An executor can answer a delivery question exactly as the policy says and pass the grader. The customer, who wanted the parcel rather than the policy, rates the contact poorly and asks again.

Use it when writing the standard at Design. Add at least one signal that comes from the customer rather than from the company: a reopened ticket, a follow-up complaint, a short rating. Read it next to the grader's verdict, band by band. The common misreading is that a high grader pass rate means satisfied customers, when it only means the company's own criteria were met.



Like a contest with two judges: *the company's grader holds up a pass, while the customer holds up their own score against what they expected.*

Liability for What the Executor Says

Whatever the executor tells a customer, the company has said. It cannot later argue that the words came from a system rather than from staff.

The clearest test so far is Moffatt v. Air Canada, decided by the British Columbia Civil Resolution Tribunal in 2024. A chatbot on the airline's website told a passenger he could claim a bereavement fare after travelling. The airline's actual policy said otherwise. The tribunal rejected the idea that the chatbot was responsible for its own statements, and ordered the airline to pay the difference.

Returns and refunds is exposed in the same way. An executor that tells a customer an opened item can still be returned within thirty days has made that the company's position for that customer, whatever the policy page says.

Use it in the rule register. Every policy statement the executor may make should trace to a signed rule, and the grader should check each answer against the rule text, not against whether it sounded helpful. The common misreading is that a disclaimer saying answers may be inaccurate shifts the risk back onto the customer.



Like a price painted in a shop window: *if the sign-writer paints the wrong price, the shop still has to honour it at the till.*

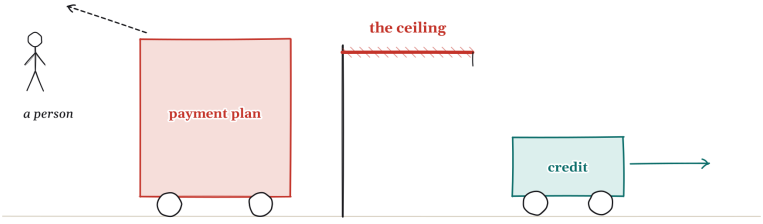
Promise Limits

An executor that talks to customers needs two kinds of limit. One covers what it may do. The other covers what it may offer, because an offer made in writing is already a kind of action.

This model is native to embedding practice. It lives in the decision register, where offers are sorted like any other decision, and in the limits set at Bound. It extends the ordinary approval limit, familiar from any delegation of authority, from money moved to words used.

Collections and cash application shows why. Sending a payment reminder is mechanical. Agreeing a payment plan or putting an account on hold is consequential and sits with the credit manager. An executor that writes "we can split this over three months" has agreed the plan, whoever was meant to approve it.

Write the list of offers the executor may make, each with a ceiling: a goodwill credit up to a set amount, a replacement but not a refund, a callback but not a delivery date. Anything above the ceiling goes to a junction. The common misreading is that limiting the executor's system permissions also limits what it can promise in a message.



Like the height bar before a low bridge: *offers that fit under the bar go through, and anything taller is stopped and sent round to a person.*

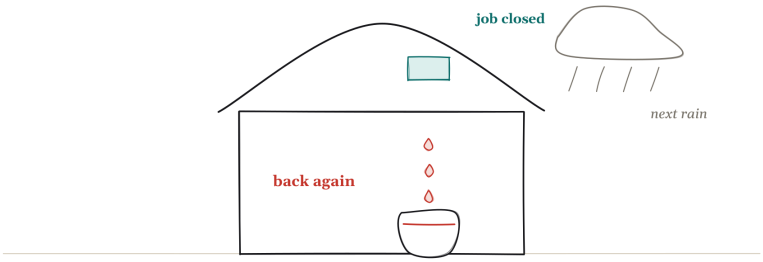
Repeat Contact as the True Measure

A case marked closed is not the same as a problem solved. The surest sign that it was not solved is that the same customer comes back about the same thing.

Bill Price, who led customer service at Amazon, made this the centre of his book with David Jaffe, The Best Service Is No Service (2008). He argued that most contacts signal a failure somewhere upstream, and that contacts per order mattered more than speed. Matthew Dixon, Karen Freeman and Nicholas Toman reached a related finding in Harvard Business Review in 2010: customer effort, including having to call again, predicted disloyalty better than delight did.

Support ticket resolution is where the trap is set. An executor can close tickets quickly and drive the closure rate up while the same customers return within days. The dashboard improves and the work grows.

Measure repeat contact on the same issue within a fixed window when you map the baseline, and carry it into the standard and the economics sheet. A repeat counts as a cost of the first contact. The common misreading is that a contact the executor deflected is a contact resolved.



Like a roof patch: *the roofer marks the job done, but the real test is whether the bucket under the ceiling fills again at the next rain.*

Service Recovery

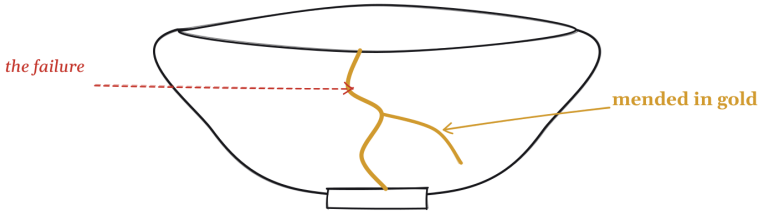
Failures will reach customers, however good the loop. What the company does next often matters more to the relationship than the failure itself. A quick, fair and visible repair can keep a customer who would otherwise have left.

Christopher Hart, James Heskett and Earl Sasser set this out in "The Profitable Art of Service Recovery" in Harvard Business Review in 1990. They argued that recovery should be planned and that front-line staff need the authority to fix things on the spot. Later research, such as Stefan Michel and

Matthew Meuter in 2008, found that recovery rarely leaves customers happier than if nothing had gone wrong.

In Complaint handling, recovery is the loop's real output. An executor can acknowledge the complaint at once, gather the facts and propose redress within its limits. It shortens the slowest part of the old process, which was usually the wait before anyone replied.

Design the recovery route as part of the loop, with remedies listed in the decision register and their ceilings set at Bound. The common misreading is that a good recovery makes up for a failure, so the failure rate matters less.



Like a bowl mended with gold: *a crack repaired quickly and visibly can keep the bowl on the table, though a bowl that never cracked is still better.*

The Peak-End Rule

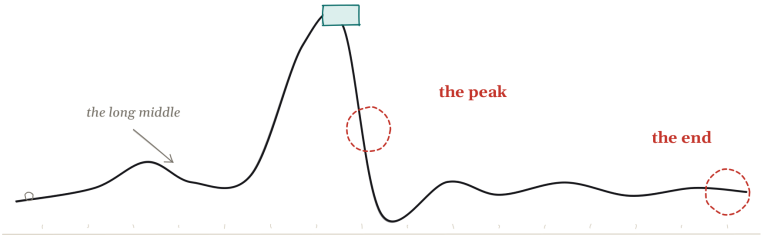
People do not remember an experience as an average of its moments. They remember mostly its most intense moment and how it ended. Its length weighs surprisingly little.

Daniel Kahneman and colleagues showed this in a 1993 paper in Psychological Science, "When More Pain Is Preferred to Less: Adding a Better End". Volunteers who held a hand in cold water for longer, with a slightly milder final stretch, preferred that trial to a shorter one. The pattern became known as the peak-end rule, and the small weight given to length as duration neglect.

Returns and refunds has an obvious peak and an obvious end. The peak is often the moment the customer is told an item is not eligible. The end is the refund arriving, or not. An executor that handles the refusal coldly and closes with a template will be remembered for those two moments, however smooth the rest.

Use it at Design and in grading. Give the refusal and the closing message particular care, and have the human audit read them first. The common

misreading is that only the end matters, so the middle of the experience can be neglected.



Like a rollercoaster ride: *people remember the steepest drop and the way it stopped, hardly how long they queued or rode.*

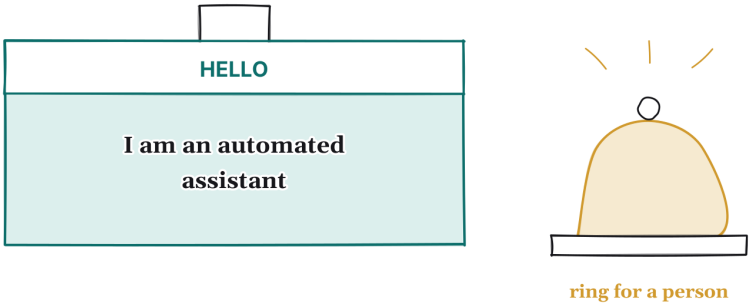
Disclosure

Customers are entitled to know when they are dealing with a machine. Telling them early is simpler than being found out late, and it lets them ask for a person when they need one.

This is now law in several places. Article 50(1) of the EU AI Act requires that people interacting with an AI system be told so, unless it is obvious. It applies from 2 August 2026, as read in 2026. California's Bolstering Online Transparency Act of 2018 already required bots used to sell to disclose that they are bots.

Lead qualification meets this first. An executor that answers a prospect's pricing questions on a website, or writes the first outreach email, is exactly the system the article describes. Disclosure becomes a design decision made once, not a review of each conversation.

Record the obligation in the rule register for every customer-facing loop. At Design, place the disclosure in the first message and add a visible path to a person, which a junction can serve. The grader can then check both on every case. The common misreading is that disclosure is a banner that satisfies the rule, when its purpose is to give the customer a real choice.



Like a name badge that says what you are dealing with, worn beside a bell that brings a person whenever the customer rings it.:

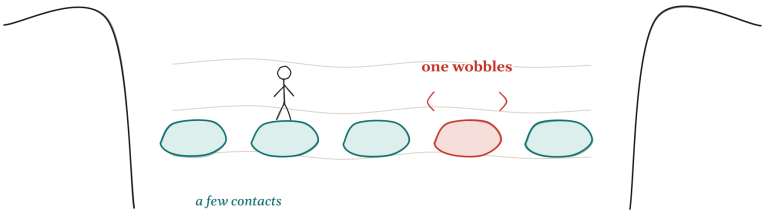
Moments of Truth

A customer meets a company in a few short contacts. Each one is a test. If it fails, the quality of everything behind it is invisible to that customer.

Jan Carlzon, who turned around the airline SAS in the early 1980s, called these contacts moments of truth in his book of that name (1987). He argued that the front line must be able to act in those moments, without passing every question up the hierarchy.

In Collections and cash application, a payment reminder is a moment of truth with a customer who is still buying. At the manufacturer, a reminder sent to a distributor who is already disputing a short delivery tells that distributor nobody has read the file. The reminder was correct and the moment was lost.

Use it at Map. Mark every point in the loop where the result reaches the customer, and check that the executor has the context it needs at each one. A reminder should not go out while a dispute is open. The common misreading is that every step in a loop is a moment of truth, when only the few the customer actually sees are.



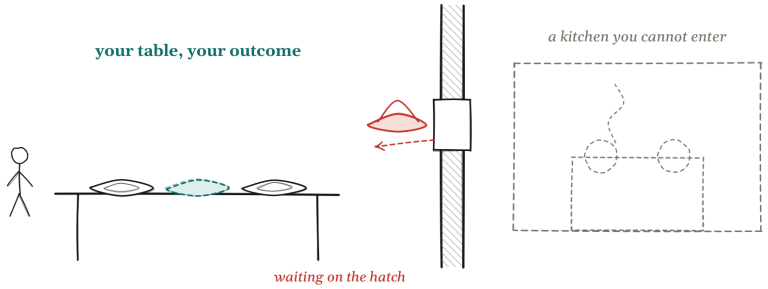
Like stepping stones across a river: *the customer touches only a few stones, and one that wobbles drops them in, however solid the rest.*

XXXVIII

Loops That Depend on Outside Parties

Some loops cannot finish inside the company. The case starts with a carrier's scan, a supplier's confirmation or a bank's file, and the action ends in someone else's system. You can recognise these loops by their waiting: the queue holds cases marked "awaiting supplier", "chasing carrier" or "pending bank". Typical examples among the reference loops are Shipment exception handling, Supplier delivery exceptions, Supplier onboarding, Payment run, Goods and services receipt and Inventory replenishment.

In these loops the company owns the outcome but not the facts. The executor reads data it did not create, acts through parties it does not manage, and pays for their mistakes as if they were its own. The models below matter most here because each one names a way an outside party can quietly decide how well the loop runs. Together they turn a vague worry about partners into questions a loop owner can answer before the executor goes live.



Like a host whose dinner depends on a neighbour's kitchen: *the host owns the meal, but the dish arrives through a hatch the host cannot see behind.*

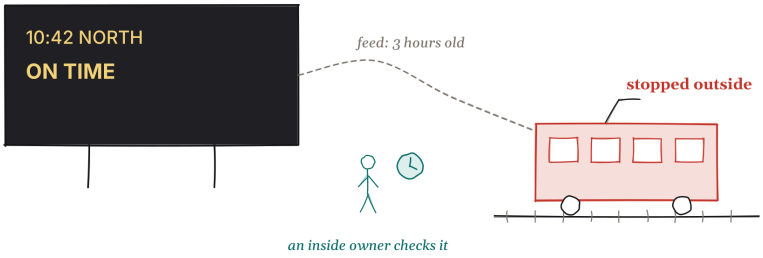
Context Keepers Outside the Company

An executor is only as good as what it reads. In these loops much of that reading is kept by someone on the other side of a contract. The carrier keeps the tracking events. The supplier keeps the order confirmation. The bank keeps the account check. Nobody inside can fix those sources, yet someone inside must answer for them.

The idea comes from embedding practice. It lives in the executor's list of context sources and keepers, extended with a column for who outside holds the source and who inside answers for it.

Shipment exception handling shows why. The executor tells a customer where an order is, and it can only repeat what the carrier's last scan says. If a carrier posts delay events hours late, the executor gives confident, wrong answers. The fault looks like the model's. It belongs to a feed nobody inside was watching.

Use it at Design. For every outside source, name an inside owner, record how fresh the data usually is, and decide what the executor does when it is stale. The common misreading is that an outside source needs no keeper because the company cannot change it.



Like a station board fed by another company's signals: *when the feed runs late the board shows confident nonsense, so someone inside must watch the feed.*

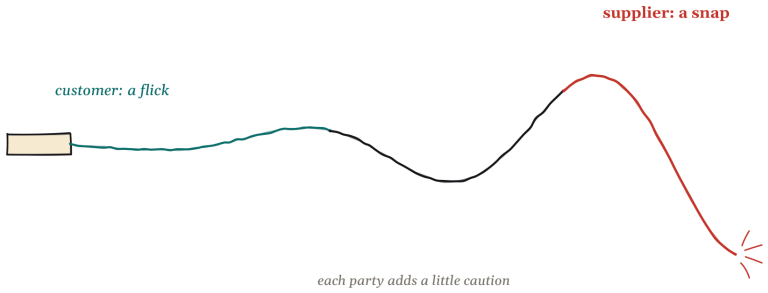
The Bullwhip Effect

Small changes at one end of a chain become large swings at the other. Each party sees only the orders of the party next to it, adds a little caution, and passes the result on. A gentle ripple in customer demand arrives at the supplier as a crashing wave.

Jay Forrester showed the pattern in simulations of production and distribution systems in Industrial Dynamics in 1961. Hau Lee, V. Padmanabhan and Seungjin Whang named it the bullwhip effect in 1997. They traced it to four causes: forecasts built on orders rather than demand, batching, price swings and rationing games.

Inventory replenishment is where an executor can make it worse. A replenishment agent that reorders faster than people did, on the same order signals, amplifies the same noise more quickly. Suppliers then see sharper swings and answer with longer lead times, which feed back into the next order.

Use it at Map, when tracing handoffs between loops. Ask what signal each loop reorders on, and feed it real demand rather than the previous loop's orders. The common misreading is that the bullwhip is a supplier problem, when the company's own ordering rules are usually the first amplifier.



Like cracking a whip: *a small flick at the customer's end of the chain grows into a violent snap by the time it reaches the supplier.*

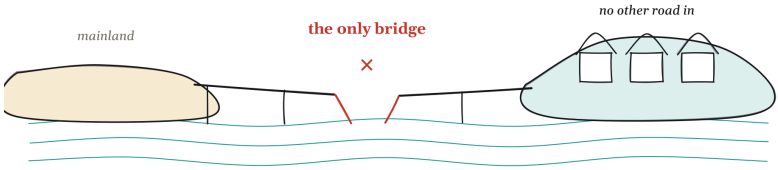
Single-Source Risk

When one outside party carries a step, its failure becomes yours. A loop with one carrier, one bank link or one supplier for a critical part looks efficient on a normal day. On a bad day it has no other road.

Peter Kraljic set out the purchasing side in the Harvard Business Review in 1983. He sorted supplies by their profit impact and supply risk, and singled out the items that are both critical and hard to source. Yossi Sheffi's The Resilient Enterprise (2005) documented how a 2000 fire at a Philips chip plant hit Ericsson, which had no alternative source, far harder than Nokia, which moved quickly to other suppliers.

Embedding adds a new single source: the data feed. Supplier delivery exceptions may run on a single visibility platform. If that platform changes its format or goes down, the executor stops reading every inbound delivery at once.

Use it in the junction register and at Bound. For each outside party, record what the loop does if it disappears for a day, and rehearse that path with people. The common misreading is that a second supplier on paper removes the risk, when the executor is still wired to only one.



Like an island village with one bridge: *it is the quickest road every day until the bridge goes, and then there is no road at all.*

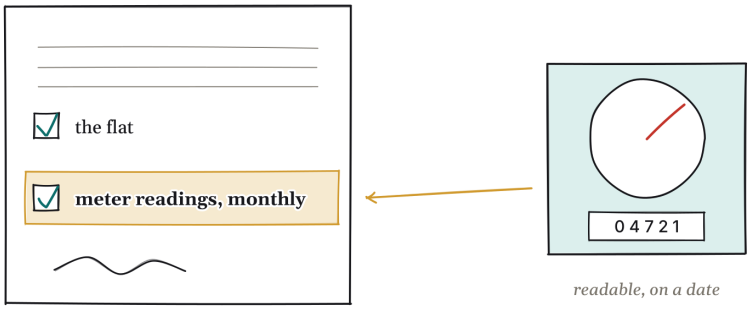
Contract the Data, Not Only the Service

Agreements with partners usually describe the service: the goods, the delivery window, the price. An executor also needs the data that comes with the service, in a form it can read and on a timetable it can rely on. If the contract is silent, the partner owes nothing when the data is late, partial or locked in a PDF.

The model comes from embedding practice and lives in the executor's list of context sources. It has a long precedent in data protection. GDPR Article 28 requires a controller to bind its processors by contract to specific duties over personal data. The same habit suits operational data.

The manufacturer met this early. Contract prices for about a third of its suppliers existed only in PDF agreements, so the invoice executor could not read them. Procurement loaded prices for the top two hundred suppliers before shadow running. Supplier onboarding is where the gap can be closed at the source.

Use it at Strip and Design. Add structured confirmations, notice periods and data formats to onboarding terms and renewals. The common misreading is that data quality is the partner's goodwill rather than a term you negotiate.



Like a flat lease that also fixes the meter readings: *write in what you will be shown and when, not only what you will get.*

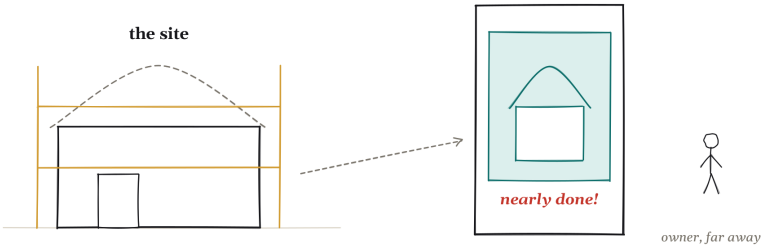
The Principal-Agent Problem

When you hire someone to act for you, their interests are close to yours but not the same. They also know things you cannot see. Left alone, they will act sensibly for themselves, and that is not always sensibly for you.

Stephen Ross framed the principal's problem in economic terms in 1973. Michael Jensen and William Meckling gave it its lasting form in 1976, as the costs of aligning and monitoring an agent whose effort the principal cannot fully observe.

Outside parties in these loops are agents in exactly this sense. A supplier eager to win an order confirms a delivery date it is unsure it can meet. A carrier marks a parcel "attempted" rather than admit a missed stop. In Supplier delivery exceptions, the executor reads these signals at face value unless someone has taught it which ones to doubt.

Use it when filling the decision register. Note where a partner's report is also a claim in its own interest, and set the executor to check that claim against an independent fact, such as the goods receipt. The common misreading is that bad partner data is careless, when it is often rational.



Like a builder sending progress photos to an owner abroad: *the report says nearly done because that keeps the payments coming, whatever the site shows.*

Lead-Time Variability

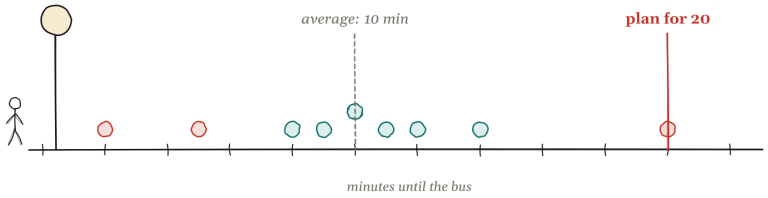
How long something takes on average matters less than how much that time wanders. A supplier who always takes six weeks is easy to plan around. One who takes four weeks or ten forces the company to hold extra stock or accept shortages, even if the average is the same.

Wallace Hopp and Mark Spearman made the point central in Factory Physics in 1996. Their laws of variability show that variation in arrivals and processing must be paid for in inventory, capacity or time. The standard safety-stock

formula in operations textbooks carries lead-time variance as its own term for the same reason.

The pattern dominates Supplier delivery exceptions. APQC's open benchmark puts median supplier on-time delivery at 90 percent, as read in 2026. That still leaves one delivery in ten off its date. An executor that re-dates exceptions quickly reduces surprise, but it cannot remove the spread.

Use it at Read. Plot the spread of each key supplier's lead times, not just the mean, and treat the wide ones as tail. The common misreading is that a faster average lead time solves the problem when the variance is unchanged.



Like a bus that comes every ten minutes on average but sometimes in two and sometimes in twenty: *you plan around the spread, not the average.*

Trust, Then Verify

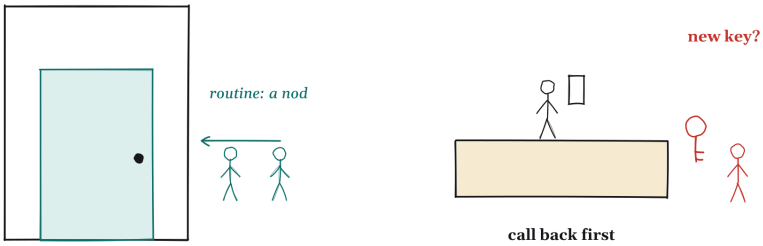
Partners must be trusted for the work to move at all. Checking every routine message from every supplier would stop the loop. The skill is to trust by default on what is cheap to undo, and to verify without exception on what is not.

The phrase adapts an old Russian proverb, "trust, but verify", which Ronald Reagan used at the signing of the Intermediate-Range Nuclear Forces Treaty in 1987. In embedding practice the idea lives in the bounded limits and the junctions. Reversibility decides where checks go.

Payments make this concrete. In the Payment run loop, a request to change a supplier's bank details is the classic route for invoice fraud. The Association for Financial Professionals' 2025 survey found that 79 percent of organisations faced attempted or actual payments fraud in 2024. A bank detail change arriving by email should never pass on trust, however familiar the sender looks.

Use it at Bound and Shadow. Let the executor accept routine confirmations alone, and route every change to bank details or remittance through a call-

back to a number already on file. The common misreading is that verification signals distrust of the partner, when it protects both sides.



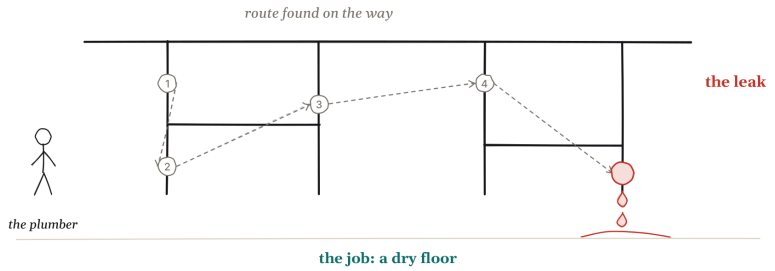
Like a doorman: *residents walk in with a nod, but anyone asking for a new key is checked by a call to the number already on file.*

XXXIX

Loops Whose Steps Are Generated

Some loops have a clear outcome but no fixed path to it. A customer reports a problem nobody has seen before, a server fails in a new way, a shipment is stuck at a border. The executor reads the case, decides what to look up, tries something, checks the result and decides again. Each case produces its own sequence of steps. You can recognise these loops because the people who run them describe their work as investigating rather than processing, and two good handlers rarely take the same route.

Typical examples among the reference loops are Support ticket resolution, IT incident management, Shipment exception handling, Supplier delivery exceptions, Complaint handling and Audit requests. With no procedure to audit, the design moves up a level: to what counts as done, what the executor may touch, and how its results are checked. The models in this chapter keep an executor that plans its own steps both useful and answerable.



Like a plumber called to a strange leak: *the job is a dry floor, and the route to the leak is found pipe by pipe on the day.*

Outcome, Fences and Trace

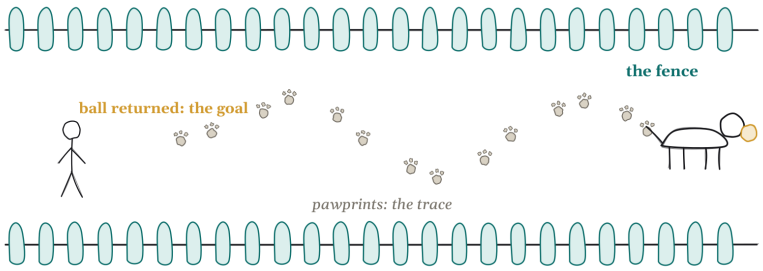
When you cannot write the steps in advance, you specify three other things instead. The first is the outcome, meaning what a finished case looks like. The second is the fences: the rules, permitted actions and junctions the executor may not cross. The third is the trace, a record of every step it actually took.

The model is native to embedding practice, where it fills the generated-steps node of the loop's canvas. Its ancestor is management by objectives, set out by

Peter Drucker in The Practice of Management in 1954. Drucker asked managers to agree results and leave the method to the person doing the work.

In Shipment exception handling, the outcome is a shipment resolved by reroute, reship, refund or claim, with the customer told accurately. The fences let the executor reroute within a cost limit but not declare a parcel lost. The trace shows which carrier records it read before choosing.

At Design, write the outcome into the standard, the fences into the rule register and the actions list, and require the platform to keep the trace for every case. The common misreading is that a trace replaces a design, when it only lets you inspect one.



Like a dog fetching a ball in a fenced garden: *you set the goal and the fence, it picks the route, and its muddy pawprints record it.*

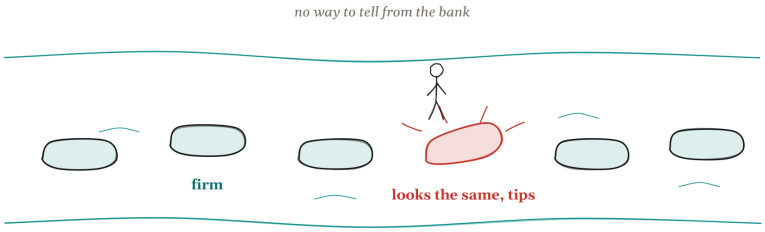
The Jagged Frontier

An AI executor is not simply good or bad at a type of work. It is excellent at some tasks and poor at neighbouring ones that look just as hard. The boundary cannot be seen from outside, and its poor answers often sound as confident as its good ones.

The phrase comes from a 2023 field experiment by Fabrizio Dell'Acqua and colleagues with consultants at Boston Consulting Group. On tasks inside the frontier, consultants using AI produced clearly better work. On a task just outside it, they did worse than colleagues without AI, because they accepted answers they should have checked.

Loops with generated steps meet the frontier case by case, since every case calls for a different chain of reasoning. In Support ticket resolution, an executor may answer order-status questions flawlessly and still misstate a refund policy. A Canadian tribunal held Air Canada liable for exactly that kind of chatbot answer in Moffatt v. Air Canada in 2024.

At Shadow, read agreement by case type and by band, never as one overall figure. Keep with people whatever types fall outside the frontier until the evidence moves. The common misreading is that a strong demonstration shows where the frontier lies.



Like stepping stones across a river: *firm stones and loose ones look identical from the bank, and the loose one tips only when trusted.*

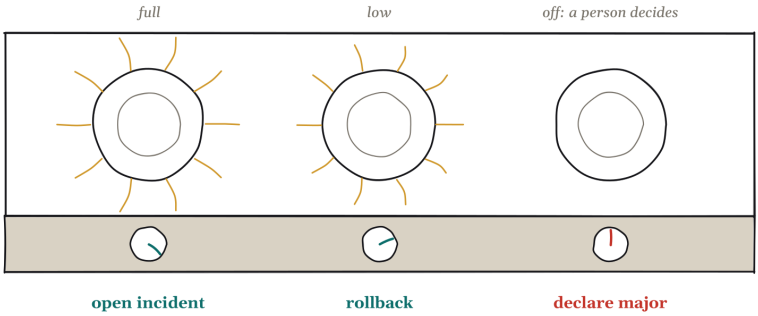
Levels of Autonomy

Handing work to a machine is not one switch between manual and automatic. There are settings in between. The machine can list options, recommend one, act once a person approves, act and then report, or act and say nothing unless asked. Each setting moves a different share of judgement away from people.

Thomas Sheridan and William Verplank set out a ten-level scale of this kind in 1978, in an MIT report on controlling undersea teleoperators. In 2000, Raja Parasuraman, Sheridan and Christopher Wickens added that the level should be chosen separately for each function of a system, not once for the whole.

That refinement matters most when steps are generated. In IT incident management, an executor might correlate alerts and open incidents on its own, apply a known rollback and report it at once, and only propose that a major incident be declared. One loop runs at three levels at the same time.

Record the level for each decision in the decision register. Raise it only through the Scale Gates, with shadow evidence for every step up. The common misreading is to give a whole loop one autonomy level, as if every decision in it carried the same risk.



Like a cooker with a knob for each ring: *one loop can run one decision on full, another on low, and keep a third switched off.*

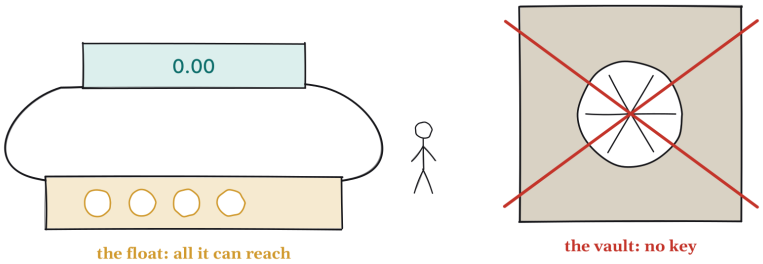
Least Privilege

Give every worker, human or machine, exactly the access its job needs and nothing more. Then a mistake, a misunderstanding or a deliberate misuse can only reach as far as that access reaches. The damage is bounded by the rights held, not by good intentions.

Jerome Saltzer and Michael Schroeder named the principle of least privilege in "The Protection of Information in Computer Systems", published in the Proceedings of the IEEE in 1975. It has been a basic rule of computer security ever since.

It bites harder when an executor plans its own steps, because nobody can list in advance every action it might attempt. An executor assembling evidence for Audit requests needs to read ledgers, populations and reconciliations. It has no need to post journal entries. Holding that right would let one wrong plan alter the very records the auditor is testing.

At Bound, compare the rights the executor holds in each system with the signed actions list, and remove everything that is not on it. Repeat the check whenever a model or tool is added. The common misreading is that broad rights are harmless as long as the instructions say not to use them.



Like a cashier's till holding only a float: *whatever goes wrong at the counter, the loss can never exceed what the drawer holds.*

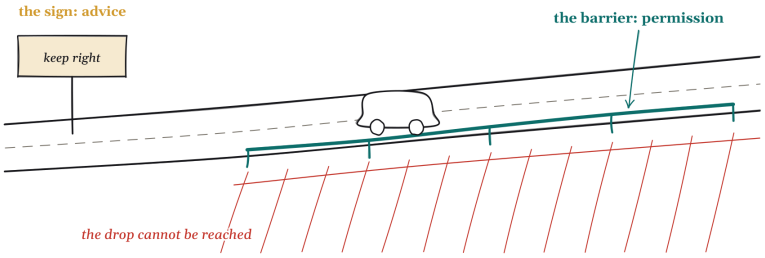
Permissions as Design

The rights an executor holds are not an administrative detail settled after the design. They are the design in its most binding form. Instructions describe what the executor should do. Permissions decide what it can do, and when the two disagree, permissions win.

The legal scholar Lawrence Lessig made the general argument in Code and Other Laws of Cyberspace in 1999. He showed that the architecture of a system regulates behaviour as firmly as law, often more firmly, because it needs no enforcement. Embedding practice applies the idea to the executor's actions list.

In Returns and refunds, policy may say that a refund outside the return window goes to a person. Written into the instructions, that sentence is advice. A permission set that lets the executor refund only inside the window, and only to the original payment method, makes the policy impossible to break. The exception reaches a person because no other path exists.

Design the permission set alongside the junctions, and treat each permission as a decision someone signs at Bound. Where a rule really matters, enforce it in permissions rather than in prompts. The common misreading is that permissions belong to IT while the business writes the rules.



Like a crash barrier on a mountain road: *a sign only asks drivers to keep right, while the barrier makes the drop impossible to reach.*

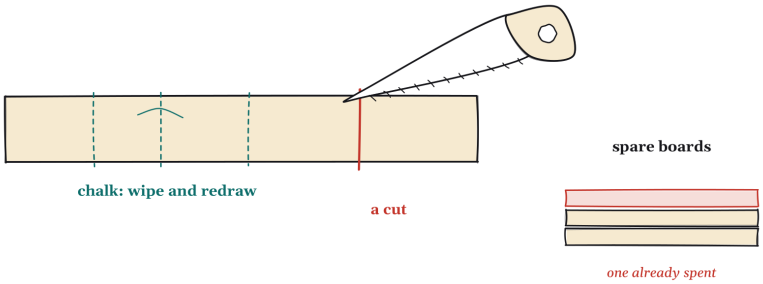
The Reversibility Budget

Some actions can be taken back cheaply, some at a cost, and some not at all. An executor that finds its own path will sometimes take a wrong step. The question is how much irreversible action it may spend before a person has to look.

The model is native to embedding practice and lives in the actions list and the economics sheet. Each action carries a cost of undoing it, and each loop receives an allowance of irreversible exposure per case and per period. Jeff Bezos drew the same line between one-way and two-way doors in Amazon's 2015 letter to shareholders.

In Supplier delivery exceptions, chasing a late supplier costs nothing to retract. Accepting a revised date can be revisited. Booking premium air freight to avoid a line stoppage at the manufacturer spends money that will not come back. A sensible budget lets the executor chase freely and expedite only up to a small sum per case, under a monthly ceiling.

Set the budget at Bound from shadow evidence, enforce it in the platform, and read its use at the operating review. The common misreading is that a reversible action is free, when undoing it still costs time and goodwill.



Like a carpenter's chalk and saw: *chalk lines can be wiped and redrawn freely, but every saw cut spends one of a few spare boards.*

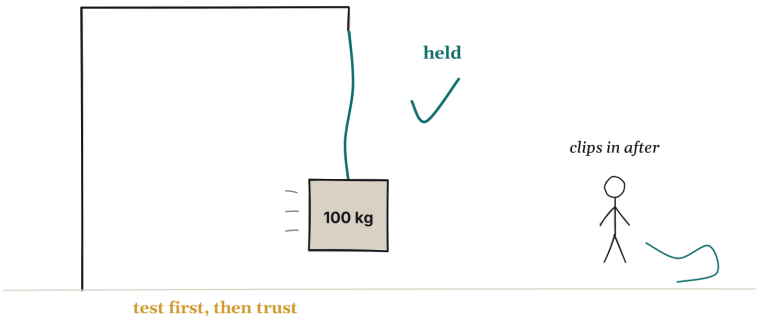
Evaluation Before Trust

Trust in a machine should follow evidence of how it performs, not run ahead of it. People tend to trust too little at first and far too much after a few good results. Both errors come from impressions rather than measurement.

John Lee and Katrina See made the argument in "Trust in Automation: Designing for Appropriate Reliance", in Human Factors in 2004. The aim, they wrote, is calibrated trust, matched to what the system can actually do. Raja Parasuraman and Victor Riley had already described the costs of misuse and disuse in 1997.

With generated steps there is no procedure to inspect, so graded outcomes are the only evidence. Klarna's experience in Support ticket resolution shows the risk. Its AI assistant handled two thirds of customer chats in its first month in 2024. The company later recruited human agents again, after its chief executive said quality had suffered.

Write the standard and build the grader before the executor touches a live case. Let autonomy grow only as graded results accumulate, at Shadow and then in Core, with a human sample checking the grader. The common misreading is that trust is earned once at launch, rather than renewed every month.



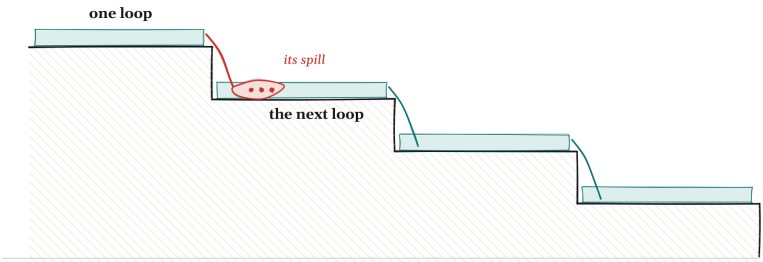
Like a new climbing rope drop-tested with a heavy weight: *the climber clips in only after the rope has held, and it is tested again each season.*

XL

Loops in a Chain

Some loops exist mainly to pass work along. Their outcome is the next loop's input, so a sales order becomes a pick list, a receipt becomes a match, and an invoice becomes a payment. You can spot them on the map by their lines. They sit inside the long chains, such as lead to cash, procure to pay and order to delivery. Typical examples are Order booking, Purchase order issue, Goods and services receipt, Supplier invoice approval, Billing and invoicing and Order fulfilment.

In a chain, one loop's tail becomes the next loop's exception. A careless receipt shows up as a failed match two loops later. Improving one loop can therefore move the problem rather than solve it. The models in this chapter keep attention on the whole chain: where it is slowest, how its parts talk to each other, and where work waits and piles up.



Like terraced fields watered from the top: *whatever one terrace lets spill, mud and all, lands in the terrace below as its problem.*

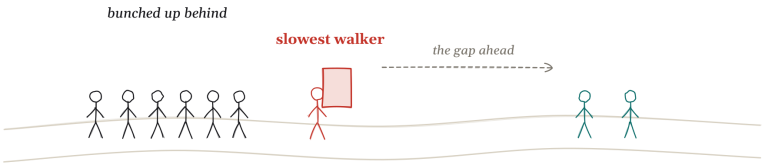
The Theory of Constraints

A chain can move no faster than its slowest link. Speeding up any other link adds nothing. It only makes the line in front of the slow one longer.

Eliyahu Goldratt set this out in The Goal (1984), a novel about a struggling plant. He gave five steps. Find the constraint, get the most from it, make everything else serve it, then add capacity to it. When it moves, start again.

In lead to cash, suppose orders spend most of their time waiting for Customer credit check. An executor that halves the time of Quote and pricing approval looks like progress. Yet the order reaches cash no sooner, because it now waits longer for a credit decision.

Use it when you Read the map and choose the next loop. Ask where cases in the chain wait longest, and start there. The common misreading is that the constraint is the busiest team, when it is the step that sets the pace of the whole chain.



Like a line of walkers on a narrow path: *the whole group arrives when the slowest walker does, and hurrying the others only bunches them behind.*

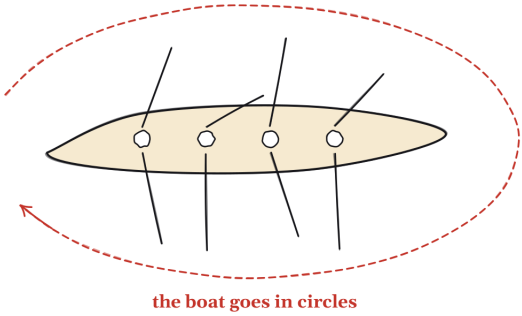
Local Optima

Each loop can do its own job well and the chain can still do badly. The best result for one part is often not the best result for the whole.

W. Edwards Deming called this suboptimisation in The New Economics (1993). He argued that a system works only when every part is managed for the aim of the whole. Goldratt made the same point in The Goal (1984). Local measures, he showed, push each department to look efficient while the plant loses money.

Consider a buyer in Purchase order issue judged on unit price. She picks the cheapest supplier, and her number improves. The cost reappears in Supplier delivery exceptions, where late and short deliveries now need chasing. No single loop report shows the trade.

At Design, write each loop's standard so that it includes what the next loop needs. Then put the chain's cost on the economics sheet beside the loop's own cost per accepted outcome. The common misreading is that local optima come from bad behaviour, when they come from honest people meeting the measures they are given.



Like a rowing crew where each rower perfects their own stroke out of time with the rest: *every oar works hard and the boat turns in circles.*

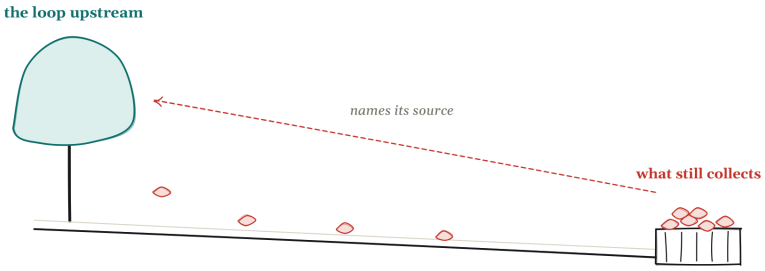
The Edges Name the Next Loop

Once a loop runs well, the cases it cannot handle stop being random. What is left tends to come from somewhere else. Read those leftovers, and they point to the loop that should go next.

This model is native to embedding practice. It lives in the map review, which reads the edges of every embedded loop before choosing what to read next. It also lives in the junction register, where each case sent to a person carries its reason.

At the manufacturer, supplier invoice approval was embedded first. Afterwards, 38% of its remaining exceptions came from missing goods receipts. The invoice loop could not fix that. The cause sat upstream, in Goods and services receipt, and the map review named that loop as a strong candidate.

Tag every exception with the loop it came from, and count the tags at each operating review. When one upstream source dominates, take it to the map review. The common misreading is that a loop′s leftover exceptions show its own limits, when they often show its neighbour′s.



Like leaves clogging a drain grate: *once the grate is clear of litter, what still collects there names the tree upstream to prune next.*

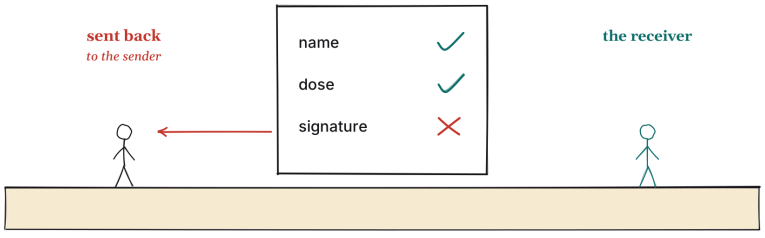
Handoffs as Contracts

When work passes between two loops, both sides need the same picture of what is being passed. Write it down as a promise. Spell out what the sender guarantees and what the receiver may assume. Say what happens when the promise is broken.

Bertrand Meyer called this design by contract in Object-Oriented Software Construction (1988). Each software routine states its preconditions and its guarantees. If a caller breaks a precondition, the fault is the caller's, and the routine does not have to guess.

Order booking hands Billing and invoicing the order lines, prices and billing triggers. If a price is missing, billing either stalls or invents one. A contract would say which fields must be present and valid at release. It would also say that an incomplete order goes back to booking.

Write the contract at Design, on the handoffs node of the Work Loop Canvas. Add its checks to the standard of both loops. The common misreading is that a contract slows the handoff, when it moves the check to the one place where the fault can be fixed cheaply.



Like a prescription at the pharmacy counter: *it is filled only when name, dose and signature are all present, and a missing field goes back to the doctor.*

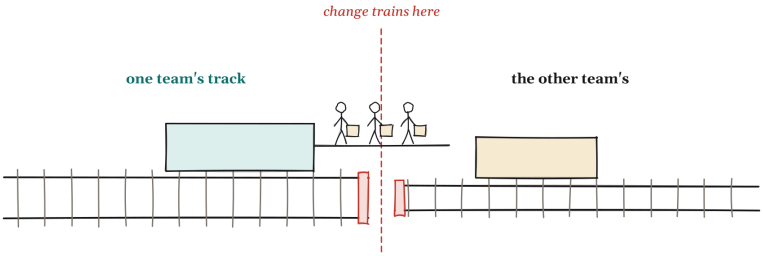
Conway's Law

Organisations tend to build systems that copy their own structure. Where two teams rarely talk, their systems meet awkwardly. The same happens to processes. The work tends to change hands wherever the company changes reporting lines.

Melvin Conway made the observation in "How Do Committees Invent?" (1968), published in Datamation. He wrote that a design mirrors the communication structure of the group that produced it. Software teams have tested and quoted the idea ever since.

At the manufacturer, order fulfilment ranked first on value but could not go first. Its rules were scattered across three plants. Each plant had its own way of handling the same order. The loop was really three loops, shaped by three management lines rather than by the order.

Use it at Map. Draw the lanes as they really are, and mark each handoff that only exists because of a reporting line. At Own, give the chain a single owner where its loops must scale together. The common misreading is that new software will fix the seams, when it usually copies them.



Like a railway that changes gauge at each border: *every country laid its own track, so passengers change trains wherever the map changes owner.*

Queues Compound

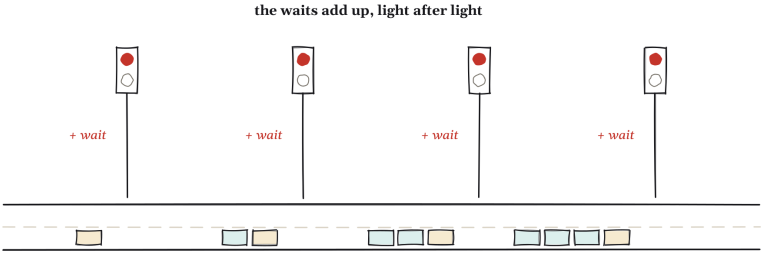
Every handoff has a queue in front of it, and the queues add up along a chain. As a team gets close to full, its waits grow much faster than its workload. Uneven arrivals make this worse, and one loop's uneven output is the next loop's uneven input.

John Little proved in 1961 that the number of cases in a system equals the arrival rate times the average time each spends inside. In the same year, John Kingman showed how waiting time rises steeply as utilisation nears one

hundred percent. Wallace Hopp and Mark Spearman, in Factory Physics (1996), traced how variability passes from station to station.

In procure to pay, a single invoice can wait at requisition approval, at the purchase order, at receipt and at matching. It then waits for the Payment run. If payments run weekly, an invoice approved just after a run waits nearly seven days more.

Use it at Strip. Add up the waits across the chain, not only inside one loop, and remove batch steps first. The common misreading is that a faster executor shortens the chain, when most of the elapsed time sits in waits between loops.



Like a road with traffic lights that are not in step: *the car stops at every one, and the small waits add up to most of the journey.*

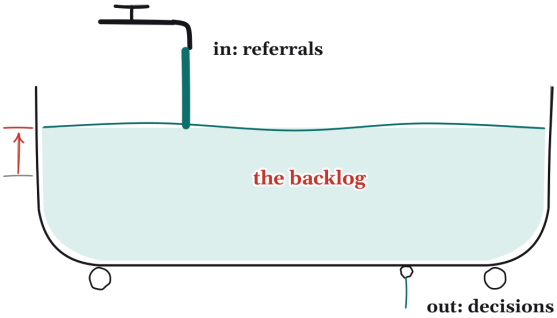
Stocks and Flows

Some things pile up and some things move. A backlog is a pile, and it changes only through what flows in and what flows out. If more arrives than leaves, the pile grows, however hard people work.

Jay Forrester built this view in Industrial Dynamics (1961). He showed that delays between stocks and flows make supply chains swing far more than demand does. John Sterman later taught the method to managers in Business Dynamics (2000).

In Supplier invoice approval, the exception queue is a stock. Cases flow in from the executor's referrals and flow out through people's decisions. At quarter-end, referrals rise while the same people are busy closing the books. The queue then grows for weeks, and payments to suppliers slip.

At Bound, set the executor's limits with the outflow in mind. Every case it refers must be one the junction staff can clear. At each operating review, watch the level of every queue between loops, not only the rate of cases handled. The common misreading is that a steady rate means a healthy loop, when a backlog can grow quietly behind it.



Like a bathtub: *the water level changes only by what the tap pours in and the drain lets out, so a faster tap fills it however busy the drain is.*

Appendix A: Areas Against Steps

The matrix shows where each knowledge area is used along the embedding lifecycle. A filled mark means the area is central to the step. An open mark means the area contributes.

Area	Read	Map	Strip	Sort	Design	Shadow	Bound	Own	Core	Scale
Distributions	●	○			○	●	●	○	○	●
Value and the map	●									●
Work and process	○	●	●	○	○				○	○
Rules and decisions		○	○	●	●	○	●		○	
The executor	○				●	●	●		○	
Control and acceptance			○	○	●	●	●	○	●	
People and accountability	○	○		○	○	○	○	●	○	●
Economics	○	●						●	●	○
Governance	○			○	○	○	●	●	●	●

Appendix B: Instruments by Step

Step	Instruments signed	Signer	Area
1. Read	Work Loop Map	Executive sponsor	Value and the map
2. Map	Process map, baseline	Process owner	Work and process; Economics
3. Strip	Strip register	Process owner	Work and process
4. Sort	Rule register, decision register, contested-decision log	Process owner; leaders for contested decisions	Rules and decisions
5. Design	Work Loop Canvas, junction register, standard	Loop owner	Control and acceptance
6. Shadow	Shadow report	Loop owner and grader	Control and acceptance
7. Bound	Gate review record	Control owner	Governance
8. Own	Loop charter, economics sheet	Loop owner; finance	People; Economics
9. Core	Operating review, switch-off plan	Loop owner	Governance; Control
10. Scale	Map review, role change plan	Executive sponsor	Value; People

Appendix C: Reading Map

Each area connects to other volumes in The Business Engineer Library, where its concepts are developed further.

Area	Where to go deeper
Distributions	*The Work Loop Canvas*, on the method of business engineering and the shape of cases
Value and the map	*The Work Loop Canvas*, on the map and the four connections to the business model
Work and process	*The Redesign*, on the passes over a documented process and what a map cannot tell you
Rules and decisions	*The Redesign*, on extracting rules and sorting decisions
The executor	The engineering volumes, on agents, context and the systems beneath them
Control and acceptance	*The Standard*, on acceptance as a business question; *Evaluation Engineering*, on grading in depth
People and accountability	*The Agentic Organization* and *The New Roles of the Agentic Era*
Economics	*AI Economics*, on the cost of intelligence across the enterprise
Governance	*The Embedding Lifecycle*, on gates and the record set step by step

Appendix D: Foundational Reading

The foundations in Part One draw on a small number of works that repay reading in full. They are listed in the order the argument uses them.

Work	Why it matters to the discipline
George Boole, *An Investigation of the Laws of Thought* (1854)	Reasoning written as algebra: the origin of the symbolic tradition
Claude Shannon, *A Symbolic Analysis of Relay and Switching Circuits* (1938)	Boole's algebra becomes the logic of machines
Kurt Gödel, *On Formally Undecidable Propositions* (1931)	No consistent rule system of sufficient power is complete
Michael Polanyi, *The Tacit Dimension* (1966)	"We can know more than we can tell": why rules in people's heads resist capture
Douglas Hofstadter, *Gödel, Escher, Bach: An Eternal Golden Braid* (1979)	Formal systems, isomorphism, levels and strange loops
Norbert Wiener, *Cybernetics* (1948)	Feedback and control in systems that act
W. Ross Ashby, *An Introduction to Cybernetics* (1956)	The law of requisite variety
Roger Conant and W. Ross Ashby, "Every Good Regulator of a System Must Be a Model of That System" (1970)	Why control requires a model
Herbert Simon, *The Sciences of the Artificial* (1969)	Design as a science of the artificial; bounded rationality
Daniel Kahneman, *Thinking, Fast and Slow* (2011)	Two modes of thought, a useful analogy for neural and symbolic components
Artur d'Avila Garcez and Luís Lamb, "Neurosymbolic AI: The 3rd Wave" (2023)	The case for combining learning and reasoning
Nassim Nicholas Taleb, *The Black Swan* (2007)	Normal and heavy-tailed worlds

Appendix E: One Loop Through Every Area

The composite manufacturer's supplier invoice loop, read through each of the nine areas. The table shows how the same loop looks from each kind of knowledge, and how the areas depend on one another.

Area	What it showed at the manufacturer
Distributions	7,400 invoices a month. 81% under €5,000 carried 14% of spend; 3% over €50,000 carried 46%. A routine body and a heavy tail
Value and the map	One of 34 loops. Value four, readiness three. Chosen first over order fulfilment, which mattered more but whose rules were scattered across three plants
Work and process	Fourteen steps in six lanes. Twenty-three minutes of work took sixteen days. Six steps struck as compensations: a strip ratio of 43%
Rules and decisions	58 rules and 9 contradictions, the largest a 2% tolerance in the system against 5% in the policy. 17 decisions: 8 mechanical, 5 bounded, 3 consequential and 1 contested, settled in thirty minutes
The executor	Contract prices for a third of suppliers existed only as PDFs; the top two hundred were loaded before shadow. A pilot account able to edit supplier records was removed
Control and acceptance	Three junctions and a tested switch-off. Shadow agreement 97.8% on the body and 81% on the tail. Bounded scope: 74% of invoices, 12% of spend
People and accountability	Named owner, junction holders, exception owner, grader and adoption owner. Nine clerks became four on the loop; three moved to supplier onboarding and one to treasury
Economics	€11.80 per invoice before; €4.10 per accepted invoice after, with the body at €2.30 and the tail at €14.60. About €680,000 a year
Governance	Gate review signed by the financial controller, with internal audit's note linking disputes to payments. Monthly operating reviews recorded a drift event and the rule it produced

The areas are not separate stages. The distribution set the bounded scope, and the contradictions explained the exceptions in the baseline. The contested decision became a junction, and the junctions set the tail's cost. Each area is a different way of reading the same loop, and the discipline is reading all nine together.

Appendix F: The Thirty-Nine Loops by Type

The table places each of the thirty-nine reference loops of *The Work Loop Library* by the types of loop in Part Seven. Every loop also draws on the models for every loop, and every loop sits in a chain. To find the models for a loop of your own, find its nearest reference loop here, then read the chapters its marks point to.

Loop	Chain	Heavy-tailed	High-volume	People	Regulated	Customer	Outside parties	Generated steps
Lead qualification	Lead to cash	●				●		●
Quote and pricing approval	Lead to cash	●			●			
Contract review	Lead to cash	●					●	●
Customer credit check	Lead to cash	●		●	●			
Order booking	Lead to cash		●					
Billing and invoicing	Lead to cash		●		●	●		
Collections and cash application	Lead to cash	●	●			●		
Supplier onboarding	Procure to pay	●					●	
Purchase requisition and approval	Procure to pay	●	●		●			
Purchase order issue	Procure to pay				●		●	
Goods and services receipt	Procure to pay		●				●	

Loop	Chain	Heavy-tailed	High-volume	People	Regulated	Customer	Outside parties	Generated steps
Supplier invoice approval	Procure to pay	●	●		●		●	
Payment run	Procure to pay	●			●		●	
Employee expense claims	Procure to pay		●	●	●			
Job requisition	Hire to retire			●				
Candidate screening	Hire to retire	●		●				
Offer and contract	Hire to retire			●	●			
Employee onboarding	Hire to retire		●	●				
Payroll changes	Hire to retire		●	●	●			
Leave and absence requests	Hire to retire		●	●				
HR helpdesk	Hire to retire		●	●		●		●
Offboarding	Hire to retire			●	●			
Journal entries	Record to report		●		●			
Account reconciliation	Record to report	●	●		●			
Intercompany settlement	Record to report	●			●			
Period close	Record to report				●			
	Record to report	●			●			

Loop	Chain	Heavy-tailed	High-volume	People	Regulated	Customer	Outside parties	Generated steps
Management and statutory reporting								
Audit requests	Record to report				●		●	
Support ticket resolution	Issue to resolution		●			●		●
Returns and refunds	Issue to resolution		●			●		
Complaint handling	Issue to resolution	●			●	●		
Warranty claims	Issue to resolution	●				●	●	
IT incident management	Issue to resolution	●			●			●
Field service dispatch	Issue to resolution			●			●	●
Order fulfilment	Order to delivery		●				●	
Inventory replenishment	Order to delivery	●					●	
Demand planning	Order to delivery						●	
Shipment exception handling	Order to delivery					●	●	●
Supplier delivery exceptions	Order to delivery	●					●	

The Five Moves

1. Put the vocabulary on one page. Take the glossary in this book and agree, with the people who will sponsor and run embedding, that these words mean these things in your company. Most arguments in the first months are about words.
2. Read one distribution. Pick one loop and chart its share of cases against its share of value. Find its body and its tail.
3. Sort one loop's decisions. List them as questions, class each one, and find the contested ones. Bring those to someone who can settle them.
4. Define your measures before you need them. Agree with finance how cost per accepted outcome will be calculated, before any loop is measured.
5. Name the roles. For your first loop, write down the person who will own it, hold its junctions, work its exceptions, grade it, own the adoption, and sign its limits.

The Mental Models

Model	Principle	Applied here
A practice becomes a discipline when it is written down	Named concepts, defined measures and shared instruments let people who never met compare and improve the work	This book writes down the concepts of embedding once
The laws of thought	Reasoning can be written as explicit rules that machines can follow	The rule and decision registers are the company's laws of thought for one loop
The limits of rules	No rule system of real power is complete	The tail, tacit knowledge and contested decisions are where rules stop
Isomorphism	Symbols mean something only when their structure mirrors the world	Registers are walked from real cases, not copied from manuals
Levels	The same system has useful laws at different levels	Choosing the level is most of abstraction
The strange loop	A system that models itself can change itself	The company models its work, acts through the model, and revises it
Neural and symbolic	Learned models are flexible without guarantees; rules are precise but brittle	An embedded loop combines both, with people at the edges
Requisite variety	Only variety can absorb variety	Rules take the body, models add variety, people hold the tail
The good regulator	Controlling a system requires a model of it	The record set is the regulator, not paperwork
Feedback	A system that senses its results can correct itself	Grade every outcome and review monthly; otherwise the loop runs open
Reflexivity	Loops change the distributions they read	Re-read the distribution after every loop runs
How distributions are made	Independent causes make normal shapes; compounding makes heavy tails	Predict the shape from the mechanism
Compression is prediction	The shortest description that still predicts is the understanding	A compression must be testable against new cases

Model	Principle	Applied here
The method loop	Read the distribution, abstract, compress, name, test, build judgement, return	Every entry is a product of the loop
Inside out	AI creates value inside processes, where companies are least documented	The body of knowledge is the inside-out complement to the business model
Body and tail	Routine body, consequential tail; design them differently	Junctions, shadow results, bounded scope and economics all split on it
Averages mislead in heavy tails	Where a few cases carry most value, the average case does not exist	Read concentration before setting any rule
Compensations	Many steps exist for constraints that are gone	Strip before writing rules
Contradictions are findings	Conflicting rules show where the process has been guessing	Each gets an owner who decides
Contested decisions	Some decisions were never agreed; only authority settles them	No loop passes Sort with one open
Decide separately from process	Decisions modelled apart can be given to an executor directly	The decision register is the executor's core brief
The jagged frontier	Executors are strong and weak in ways not visible in advance	Shadow reads agreement case by case and band by band
The junction	People decide where actions are costly or irreversible	Placed by reversibility, with evidence and a way back
The ironies of automation	Supervisors of automation over-trust it and lose skills	Override rates are health signals; the tail must be learned deliberately
Per accepted outcome	Cost means fully loaded cost over results that met the standard	The only economic measure reported
Goodhart's warning	A measure made into a target stops measuring	Activity counts are kept out of reports
Scale follows signatures	Readiness is shown by signed documents, not confidence	The Scale Gates open on signatures

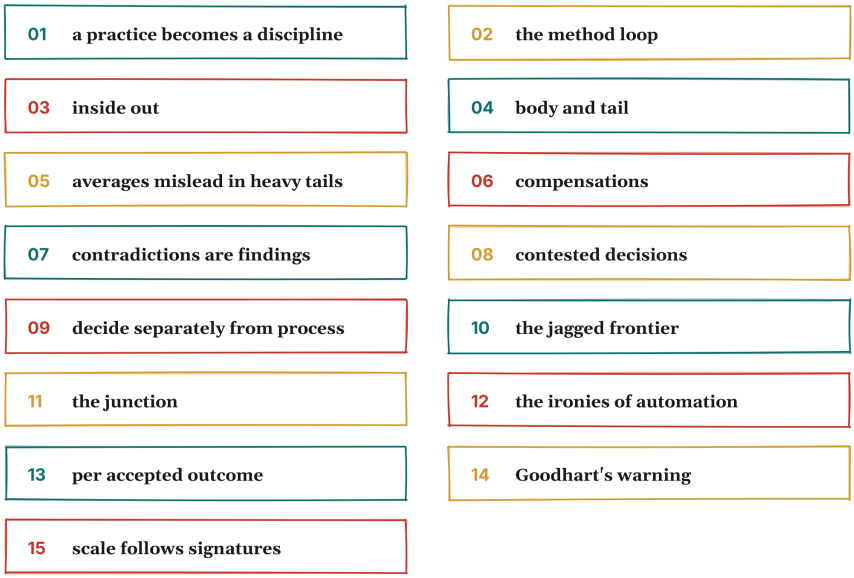


FIGURE 30 The mental models of the body of knowledge.

Glossary

Term	Definition
Abstraction	Seeing the few kinds of case behind the many, deliberately leaving out detail
Accretion	The layer of compensations around a loop's genuine steps
Actions	The changes an executor may make, each with its system and whether it can be undone
Adoption owner	The person responsible for people whose work changes and for how new staff learn
Agreement	The share of cases where the executor's result matched the accepted result, reported for body and tail
Architect	A business engineer who holds the method for a company
Baseline	Measured volume, fully loaded cost per case, elapsed time, exception rate and first-time-right before change
Body	The large share of routine, similar cases in a loop
Bounded decision	A decision needing judgement inside a written limit; the executor may take it alone
Bounded scope	The signed conditions under which the executor acts alone
Business engineer	The practitioner who runs embedding lifecycles
Business engineering	The practice of understanding how a business actually works, well enough to redesign it
Case	One unit of work handled by a loop from start to finish
Chain	A sequence of loops in which each one's outcome is the next one's input
Closed loop	A system that acts, senses the result, compares it with a goal and adjusts
Compensation	A step that exists only to work around a constraint
Compression	Stating a mechanism in the fewest words that still predict something
Concentration	The share of value carried by the top 5% of cases by value
Consequential decision	A decision with a high cost of error or an irreversible action; a person decides
Contested decision	A decision the organisation never agreed on; settled by authority in writing
Contested-decision log	The record of contested decisions, positions, settler, answer and date

Term	Definition
Context	The information an executor must read to act well on a case
Contradiction	Two rules that cannot both be followed
Control owner	The person responsible for controls over a process; signs bounded limits
Cost per accepted outcome	Fully loaded cost divided by outcomes meeting the standard
Decider	An executor given an outcome, rules and limits, which works out its own steps
Decision register	The list of a loop's decisions, with class, reason and limit
Decision rights	Who may take which decision, up to which limit
Decision	Any point where the work could go more than one way
Distribution shift	A change in how cases spread, which can move cases from body to tail
Distribution	How a loop's cases spread by value, difficulty, risk or time
Drift	A gradual change in the quality of outcomes as inputs, models or people change
Economics sheet	The signed record of cost per accepted outcome against the baseline
Elapsed time	Time from a case's arrival to its accepted outcome
Embedding	An executor that decides running part of a core process, inside written limits, with named people answering for it
Event log	Time-stamped system records of the steps each case went through
Exception owner	The person who works the exception queue
Exception queue	The single place for cases outside every rule, with owner and time limit
Executive sponsor	The senior person who commissions embedding and settles cross-department disagreements
Executor	Whatever does the work in a loop
Feedback	Information about the effect of an action, used to adjust the next action
First-time-right	Cases accepted without rework, as a share of all cases
Flow efficiency	Hands-on time divided by elapsed time
Fully loaded cost	All costs of a loop, including people at real cost, systems, rework and errors
Gate review record	The signed record of the decision at Bound
Gate	

Term	Definition
	The end of a lifecycle step, where a named person decides whether work moves on
Good regulator	The principle that controlling a system well requires a model of it
Grader	The person who keeps the standard and grading calibrated
Grading	Applying the standard to every outcome, by rules, a model grader and a human audit
Handoff	A point where a case moves between lanes or between loops
Heavy-tailed distribution	A spread in which a few cases carry most of the value or risk
Instruction-follower	An executor that needs its steps specified in advance
Instructions	The executor's brief, drawn from the signed registers
Isomorphism	A structural correspondence that lets symbols mean something in the world
Jagged frontier	The uneven boundary of tasks an executor does well and badly
Junction holder	The named person who decides at a junction
Junction	A point where a person decides instead of the executor, with defined evidence and reversal
Lead	A business engineer who runs lifecycles end to end
Loop charter	The signed list of named people in every loop role, and the agreed changes to work
Loop owner	The person who answers for a loop's outcome
Map review	The quarterly update of the Work Loop Map
Mechanical decision	A decision that follows a written rule completely
Method loop	Read the distribution, abstract, compress, name, test, build judgement, return
Model	The language model at the centre of an executor
Neural model	A system that learns patterns from examples rather than following written rules
Neurosymbolic system	A system that combines learned models with explicit rules, tools and verification
Normal distribution	A spread in which cases cluster around a typical value
Operating review	The monthly review of a loop in the core, ending in recorded decisions

Term	Definition
Outcome	The result a loop produces, stated so an outsider can accept or reject it
Override rate	How often junction holders overrule the executor's evidence
Permissions	The rights an executor holds in each system, matched to its actions
Pilot	A test of whether an executor can do a task on prepared cases
Platform	The layer that runs the loop: routing, limits, logs, switch-off, grader access
Practitioner	A business engineer who runs lifecycle steps under guidance
Preferential attachment	A mechanism in which what is large grows faster, producing heavy tails
Process map	A swimlane drawing of how a loop runs today
Process mining	Reconstructing the real flow of work from event logs
Process owner	The person responsible for how a process runs today
Readiness	A 1 to 5 score of how close a loop is to being embedded
Record set	The full collection of a loop's signed documents
Reflexivity	The way a loop's actions change the distribution it reads
Requisite variety	The principle that a controller needs as many responses as the disturbances it must handle
Rework	A step that redoes what should have been right the first time
Role change plan	The map-wide plan for how roles and expertise change
Rule register	The list of a loop's rules with source, owner and date
Rule	A statement of what must or must not happen in a case
Scale Gates	G0 Mapped, G1 Shadow, G2 Bounded, G3 Owned, G4 Core
Scope change	Any widening of what the executor may do alone
Shadow running	The executor working real cases beside people without acting
Signature	A named person's dated, recorded acceptance of a document
Standard	The written definition of an accepted outcome and how it is graded
Strange loop	A system that, moving through its levels, comes back to represent and change itself
Strip ratio	Steps struck as compensations, divided by steps on the signed map
Switch-off	The tested means of stopping a loop and returning cases to people

Term	Definition
Symbolic layer	The rules, decisions, limits and standards a business writes down for a loop
Tacit knowledge	Knowledge people use but cannot fully state
Tail	The small share of cases carrying most of the value, risk or difficulty
Topology	The steps an outcome genuinely requires
Value to the core	A 1 to 5 score of how much a loop matters to the business
Walk-through	Following real cases through a process map
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 and readiness
Work loop	The unit that produces one outcome end to end

Notes and Sources

Laws of thought and computation. George Boole, *An Investigation of the Laws of Thought* (1854). Claude E. Shannon, "A Symbolic Analysis of Relay and Switching Circuits," *Transactions of the AIEE* 57 (1938), from his 1937 master's thesis. J. McCarthy, M. Minsky, N. Rochester and C. Shannon, *A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence* (1955).

Limits of formal systems. Kurt Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I" (1931). Alan Turing, "On Computable Numbers, with an Application to the Entscheidungsproblem" (1936). Michael Polanyi, *The Tacit Dimension* (1966). On expert systems: Edward Shortliffe, *Computer-Based Medical Consultations: MYCIN* (1976); John McDermott, "R1: A Rule-Based Configurer of Computer Systems," *Artificial Intelligence* 19 (1982); Edward Feigenbaum on the knowledge acquisition bottleneck.

Strange loops. Douglas R. Hofstadter, *Gödel, Escher, Bach: An Eternal Golden Braid* (1979) and *I Am a Strange Loop* (2007).

The neural tradition. Warren McCulloch and Walter Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biophysics* 5 (1943). Frank Rosenblatt, "The Perceptron," *Psychological Review* 65 (1958). Marvin Minsky and Seymour Papert, *Perceptrons* (1969). David Rumelhart, Geoffrey Hinton and Ronald Williams, "Learning Representations by Back-Propagating Errors," *Nature* 323 (1986). Grégoire Delétang et al., "Language Modeling Is Compression," ICLR (2024).

Neurosymbolic AI. Daniel Kahneman, *Thinking, Fast and Slow* (2011). Henry Kautz, "The Third AI Summer," AAAI Robert S. Engelmore Memorial Lecture (2020), published in *AI Magazine* (2022). Artur d'Avila Garcez and Luís C. Lamb, "Neurosymbolic AI: The 3rd Wave," *Artificial Intelligence Review* (2023). Gary Marcus, "The Next Decade in AI: Four Steps Towards Robust Artificial Intelligence" (2020). Trieu Trinh et al., "Solving Olympiad Geometry Without Human Demonstrations," *Nature* 625 (2024).

Cybernetics and control. Norbert Wiener, *Cybernetics: Or Control and Communication in the Animal and the Machine* (1948). W. Ross Ashby, *An Introduction to Cybernetics* (1956). 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). George Soros, *The Alchemy of Finance* (1987), on reflexivity.

Distributions and models. Robert K. Merton, "The Matthew Effect in Science," *Science* 159 (1968). Albert-László Barabási and Réka Albert, "Emergence of Scaling in Random Networks," *Science* 286 (1999). Alfred Korzybski, *Science and Sanity* (1933). George E. P. Box, "Science and Statistics," *Journal of the American Statistical Association* 71 (1976). Karl Popper, *Logik der Forschung* (1934). Herbert A. Simon, *The Sciences of the Artificial* (1969). On compression: Andrey Kolmogorov (1965) and Jorma Rissanen, "Modeling by Shortest Data Description," *Automatica* 14 (1978).

Double-entry bookkeeping. Luca Pacioli, *Summa de arithmetica, geometria, proportioni et proportionalità* (Venice, 1494), which contains the treatise on bookkeeping known as *Particularis de computis et scripturis*.

The Business Model Canvas. Alexander Osterwalder and Yves Pigneur, *Business Model Generation* (2010).

Heavy tails. Vilfredo Pareto's observations on income concentration in *Cours d'économie politique* (1896–97). Nassim Nicholas Taleb, *The Black Swan* (2007), for "Mediocristan" and "Extremistan".

Intuition. Daniel Kahneman and Gary Klein, "Conditions for Intuitive Expertise: A Failure to Disagree," *American Psychologist* 64, no. 6 (2009).

Process redesign and mining. Michael Hammer, "Reengineering Work: Don't Automate, Obliterate," *Harvard Business Review* (July–August 1990). Wil M. P. van der Aalst, *Process Mining: Data Science in Action*, 2nd ed. (2016). Niklas Modig and Pär Åhlström, *This Is Lean* (2012), on flow efficiency.

Decisions. Object Management Group, *Decision Model and Notation (DMN) 1.0* (2015).

The stop cord. Taiichi Ohno, *Toyota Production System: Beyond Large-Scale Production* (English edition, 1988).

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

The jagged frontier. Fabrizio Dell'Acqua et al., "Navigating the Jagged Technological Frontier," Harvard Business School Working Paper 24-013 (2023).

Productivity and learning. Erik Brynjolfsson, Danielle Li and Lindsey Raymond, "Generative AI at Work," *Quarterly Journal of Economics* 140, no. 2 (2025).

Automation. Lisanne Bainbridge, "Ironies of Automation," *Automatica* 19, no. 6 (1983). Raja Parasuraman and Victor Riley, "Humans and Automation: Use, Misuse, Disuse, Abuse," *Human Factors* 39, no. 2 (1997).

Measures. Charles Goodhart, "Problems of Monetary Management: The U.K. Experience" (1975), the source of Goodhart's law.

Adoption. McKinsey & Company, *The state of AI in 2025: Agents, innovation, and transformation* (November 2025). MIT NANDA, *The GenAI Divide: State of AI in Business 2025* (2025), as summarised in Virtualization Review, 19 August 2025.

Governance. National Institute of Standards and Technology, *AI Risk Management Framework (AI RMF 1.0)* (January 2023). ISO/IEC 42001:2023. Regulation (EU) 2024/1689 (the AI Act). *Moffatt v. Air Canada*, 2024 BCCRT 149.

The composite manufacturer. Invoice figures are set within ranges reported by Ardent Partners, *Accounts Payable Metrics that Matter in 2025*, and APQC Open Standards Benchmarking.

Models for Every Loop. Michael Hammer and James Champy, *Reengineering the Corporation: A Manifesto for Business Revolution* (1993). Vilfredo Pareto, *Cours d'économie politique* (1896). Nassim Nicholas Taleb, *The Black Swan: The Impact of the Highly Improbable* (2007). Michael Hammer, "Reengineering Work: Don't Automate, Obliterate," *Harvard Business Review* (1990). Herbert A. Simon, *The New Science of Management Decision* (1960). Kurt Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I," *Monatshefte für Mathematik und Physik* (1931). Taiichi Ohno, *Toyota Production System: Beyond Large-Scale Production* (1978; English edition 1988). Thomas B. Sheridan and William L. Verplank, *Human and Computer Control of Undersea Teleoperators*, MIT Man-Machine Systems Laboratory (1978). Regulation (EU) 2016/679, General Data Protection Regulation, Article 22 (2016). W.

Ross Ashby, *An Introduction to Cybernetics* (1956). 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* (1970). Walter A. Shewhart, *Economic Control of Quality of Manufactured Product* (1931).

Norbert Wiener, *Cybernetics: Or Control and Communication in the Animal and the Machine* (1948). Philip B. Crosby, *Quality Is Free* (1979). Robert G. Cooper, "Stage-Gate Systems: A New Tool for Managing New Products," *Business Horizons* (1990). Regulation (EU) 2024/1689, Artificial Intelligence Act, Annex III (2024). George Soros, *The Alchemy of Finance* (1987). Robert K. Merton, "The Self-Fulfilling Prophecy," *The Antioch Review* (1948).

Heavy-Tailed Loops. Vilfredo Pareto, *Cours d'économie politique* (1896). Benoit Mandelbrot, "The Variation of Certain Speculative Prices", *Journal of Business* 36 (1963). Nassim Nicholas Taleb, *The Black Swan: The Impact of the Highly Improbable* (2007). Robert K. Merton, "The Matthew Effect in Science", *Science* 159 (1968). Albert-László Barabási and Réka Albert, "Emergence of Scaling in Random Networks", *Science* 286 (1999). Sam L. Savage, *The Flaw of Averages: Why We Underestimate Risk in the Face of Uncertainty* (2009). Joseph M. Juran, *Quality Control Handbook* (1951). H. Ford Dickie, "ABC Inventory Analysis Shoots for Dollars, Not Pennies", *Factory Management and Maintenance* (1951). Peter Kraljic, "Purchasing Must Become Supply Management", *Harvard Business Review* (1983). Albert O.

Hirschman, *National Power and the Structure of Foreign Trade* (1945). Orris C. Herfindahl, *Concentration in the Steel Industry*, doctoral dissertation, Columbia University (1950). Basel Committee on Banking Supervision, *Supervisory Framework for Measuring and Controlling Large Exposures* (2014). Abraham Wald, *A Method of Estimating Plane Vulnerability Based on Damage of Survivors*, Statistical Research Group memoranda (1943), reprinted by the Center for Naval Analyses (1980). Frédéric Bastiat, *Ce qu'on voit et ce qu'on ne voit pas* (1850). Daniel Bernoulli, "Specimen theoriae novae de mensura sortis", *Commentarii Academiae Scientiarum Imperialis Petropolitanae* (1738). Philippe Artzner, Freddy Delbaen, Jean-Marc Eber and David Heath, "Coherent Measures of Risk", *Mathematical Finance* 9 (1999).

High-Volume Routine Loops. Niklas Modig and Pär Åhlström, *This Is Lean: Resolving the Efficiency Paradox* (2012). George Stalk Jr. and Thomas M. Hout, *Competing Against Time: How Time-Based Competition Is Reshaping Global Markets* (1990). John D. C. Little, "A Proof for the Queuing Formula: L = λW", *Operations Research* 9(3) (1961). Taiichi Ohno, *Toyota Production System: Beyond Large-Scale Production* (1978; English edition 1988). Walter A. Shewhart, *Economic Control of Quality of Manufactured Product* (1931). W. Edwards Deming, *Out of the Crisis* (1986). Gerhard Widmer and Miroslav

Kubat, "Learning in the Presence of Concept Drift and Hidden Contexts", *Machine Learning* 23 (1996). German Growth Opportunities Act (Wachstumschancengesetz), B2B e-invoicing receipt obligation from 1 January 2025 (as read in 2026).

Lisanne Bainbridge, "Ironies of Automation", *Automatica* 19(6) (1983). Charles A. E. Goodhart, "Problems of Monetary Management: The U.K. Experience", *Papers in Monetary Economics*, Reserve Bank of Australia (1975). Marilyn Strathern, "'Improving Ratings': Audit in the British University System", *European Review* 5(3) (1997). T. P. Wright, "Factors Affecting the Cost of Airplanes", *Journal of the Aeronautical Sciences* 3(4) (1936). Boston Consulting Group, *Perspectives on Experience* (1968).

Loops That Decide About People. European Parliament and Council, *Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act)*, Annex III and Article 14 (2024). Colorado General Assembly, *Senate Bill 24-205, Consumer Protections for Artificial Intelligence* (2024). European Parliament and Council, *Regulation (EU) 2016/679 (General Data Protection Regulation)*, Articles 5, 9, 22 and 35 (2016). Court of Justice of the European Union, *Case C-634/21, OQ v Land Hessen (SCHUFA Holding)* (2023). US Equal Employment Opportunity Commission and others, *Uniform Guidelines on Employee Selection Procedures*, 29 CFR Part 1607 (1978). US Supreme Court, *Griggs v. Duke Power Co.*, 401 U.S. 424 (1971). New York City Council, *Local Law 144 of 2021 on automated employment decision tools* (2021).

Solon Barocas and Andrew D. Selbst, *Big Data's Disparate Impact*, California Law Review 104 (2016). Anya E. R. Prince and Daniel Schwarcz, *Proxy Discrimination in the Age of Artificial Intelligence and Big Data*, Iowa Law Review 105 (2020). Jeffrey Dastin, *Amazon scraps secret AI recruiting tool that showed bias against women*, Reuters (2018). *Equal Credit Opportunity Act*, 15 U.S.C. 1691 (1974), and *Regulation B*, 12 CFR 1002.9. Sandra Wachter, Brent Mittelstadt and Chris Russell, *Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR*, Harvard Journal of Law and Technology 31 (2018). OECD, *Guidelines on the Protection of Privacy and Transborder Flows of Personal Data* (1980).

Raja Parasuraman and Victor Riley, *Humans and Automation: Use, Misuse, Disuse, Abuse*, Human Factors 39 (1997). Linda J. Skitka, Kathleen L. Mosier and Mark Burdick, *Does Automation Bias Decision-Making?*, International Journal of Human-Computer Studies 51 (1999). Marco Almada, *Human*

Intervention in Automated Decision-Making: Toward the Construction of Contestable Systems, Proceedings of the Seventeenth International Conference on Artificial Intelligence and Law (2019). European Parliament and Council, *Directive (EU) 2024/2831 on improving working conditions in platform work* (2024).

Regulated and Audited Loops. Committee of Sponsoring Organizations of the Treadway Commission, *Internal Control: Integrated Framework* (1992; updated 2013). *Sarbanes-Oxley Act of 2002*, Public Law 107-204, sections 302 and 404 (2002). Luca Pacioli, *Summa de arithmetica, geometria, proportioni et proportionalita* (1494). Public Company Accounting Oversight Board, *AS 1105: Audit Evidence* (2010). Institute of Internal Auditors, *The Three Lines of Defense in Effective Risk Management and Control* (2013). Institute of Internal Auditors, *The IIA's Three Lines Model: An Update of the Three Lines of Defense* (2020). International Accounting Standards Board, *Definition of Material: Amendments to IAS 1 and IAS 8* (2018). US Securities and Exchange Commission, *Staff Accounting Bulletin No. 99: Materiality* (1999).

International Auditing and Assurance Standards Board, *ISA 500: Audit Evidence* (2009). ISACA, *COBIT: Control Objectives for Information and Related Technologies* (1996). American Institute of Certified Public Accountants, *SSAE No. 18: Attestation Standards: Clarification and Recodification* (2017). Jeff Bezos, *2015 Letter to Shareholders*, Amazon.com (2016). Regulation (EU) 2024/886 of the European Parliament and of the Council on instant credit transfers in euro (2024).

Customer-Facing Loops. A. Parasuraman, Valarie A. Zeithaml and Leonard L. Berry, "A Conceptual Model of Service Quality and Its Implications for Future Research", *Journal of Marketing* 49(4) (1985). A. Parasuraman, Valarie A. Zeithaml and Leonard L. Berry, "SERVQUAL: A Multiple-Item Scale for Measuring Consumer Perceptions of Service Quality", *Journal of Retailing* 64(1) (1988). *Moffatt v. Air Canada*, 2024 BCCRT 149, British Columbia Civil Resolution Tribunal (2024). Bill Price and David Jaffe, *The Best Service Is No Service* (2008). Matthew Dixon, Karen Freeman and Nicholas Toman, "Stop Trying to Delight Your Customers", *Harvard Business Review* (2010). Christopher W. L. Hart, James L. Heskett and W. Earl Sasser Jr., "The Profitable Art of Service Recovery", *Harvard Business Review* (1990).

Stefan Michel and Matthew L. Meuter, "The Service Recovery Paradox: True but Overrated?", *International Journal of Service Industry Management* 19(4) (2008). Daniel Kahneman, Barbara L. Fredrickson, Charles A. Schreiber

and Donald A. Redelmeier, "When More Pain Is Preferred to Less: Adding a Better End", *Psychological Science* 4(6) (1993). Regulation (EU) 2024/1689 (Artificial Intelligence Act), Article 50(1) (2024). California Senate Bill 1001, Bolstering Online Transparency Act, Business and Professions Code sections 17940 to 17943 (2018). Jan Carlzon, *Moments of Truth* (1987).

Loops That Depend on Outside Parties. Jay W. Forrester, *Industrial Dynamics* (1961). Hau L. Lee, V. Padmanabhan and Seungjin Whang, "Information Distortion in a Supply Chain: The Bullwhip Effect", *Management Science* 43(4) (1997). Peter Kraljic, "Purchasing Must Become Supply Management", *Harvard Business Review* (1983). Yossi Sheffi, *The Resilient Enterprise: Overcoming Vulnerability for Competitive Advantage* (2005). Regulation (EU) 2016/679, *General Data Protection Regulation*, Article 28 (2016). Stephen A. Ross, "The Economic Theory of Agency: The Principal's Problem", *American Economic Review* 63(2) (1973). Michael C. Jensen and William H. Meckling, "Theory of the Firm: Managerial Behavior, Agency Costs and Ownership Structure", *Journal of Financial Economics* 3(4) (1976). Wallace J. Hopp and Mark L. Spearman, *Factory Physics* (1996).

APQC, *Open Standards Benchmarking*, measure 102645, "Percentage of supplier on-time delivery" (accessed 2026). Ronald Reagan, *Remarks on Signing the Intermediate-Range Nuclear Forces Treaty* (1987). Association for Financial Professionals, *2025 AFP Payments Fraud and Control Survey* (2025).

Loops Whose Steps Are Generated. Peter F. Drucker, *The Practice of Management* (1954). Fabrizio Dell'Acqua et al., "Navigating the Jagged Technological Frontier: Field Experimental Evidence of the Effects of AI on Knowledge Worker Productivity and Quality," Harvard Business School Working Paper 24-013 (2023). *Moffatt v. Air Canada*, 2024 BCCRT 149, Civil Resolution Tribunal of British Columbia (2024). Thomas B. Sheridan and William L. Verplank, *Human and Computer Control of Undersea Teleoperators*, MIT Man-Machine Systems Laboratory technical report (1978). Raja Parasuraman, Thomas B. Sheridan and Christopher D. Wickens, "A Model for Types and Levels of Human Interaction with Automation," *IEEE Transactions on Systems, Man, and Cybernetics, Part A* 30, no. 3 (2000). Jerome H. Saltzer and Michael D.

Schroeder, "The Protection of Information in Computer Systems," *Proceedings of the IEEE* 63, no. 9 (1975). Lawrence Lessig, *Code and Other Laws of Cyberspace* (1999). Jeff Bezos, "2015 Letter to Shareholders,"

Amazon.com (2016). John D. Lee and Katrina A. See, "Trust in Automation: Designing for Appropriate Reliance," *Human Factors* 46, no. 1 (2004). Raja Parasuraman and Victor Riley, "Humans and Automation: Use, Misuse, Disuse, Abuse," *Human Factors* 39, no. 2 (1997). OpenAI, "Klarna," customer story, openai.com (2024). Entrepreneur, "Klarna CEO Reverses Course by Hiring More Humans, Not AI" (2025).

Loops in a Chain. Eliyahu M. Goldratt and Jeff Cox, *The Goal: A Process of Ongoing Improvement* (1984). W. Edwards Deming, *The New Economics for Industry, Government, Education* (1993). Bertrand Meyer, *Object-Oriented Software Construction* (1988). Melvin E. Conway, "How Do Committees Invent?", *Datamation* (1968). John D. C. Little, "A Proof for the Queuing Formula: L = λW", *Operations Research* (1961). J. F. C. Kingman, "The Single Server Queue in Heavy Traffic", *Proceedings of the Cambridge Philosophical Society* (1961). Wallace J. Hopp and Mark L. Spearman, *Factory Physics* (1996). Jay W. Forrester, *Industrial Dynamics* (1961). John D. Sterman, *Business Dynamics: Systems Thinking and Modeling for a Complex World* (2000).

The Thread

This book threads into The Business Engineer Library through the instruments every volume shares. The Work Loop Canvas and the Work Loop Map describe a loop and a company's loops. The junction, the charter and the standard keep people answerable for an executor's work. The drift check watches quality in the core. Each is defined once here, in its area, and developed further in the volumes listed in Appendix C.

Also in This Series

This book is the first in a series on embedding AI into the enterprise core, and the one the others follow from. Each volume stands alone, and together they form one discipline.

The Business Engineering Body of Knowledge. This book: the foundations, the core method, and every concept, measure and instrument of the discipline, defined once.

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

The Embedding Lifecycle. The sequence: ten steps from the first reading of a company's loops to an executor running in the core, with what is signed at each.

The Work Loop Library. The record: thirty-nine reference loops across six chains, six of them taken through the full lifecycle with every template filled in.

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

THE BUSINESS ENGINEER

The Business Engineering Body of Knowledge

Read the distribution. Write the rules. Close the loop.

businessengineer.ai